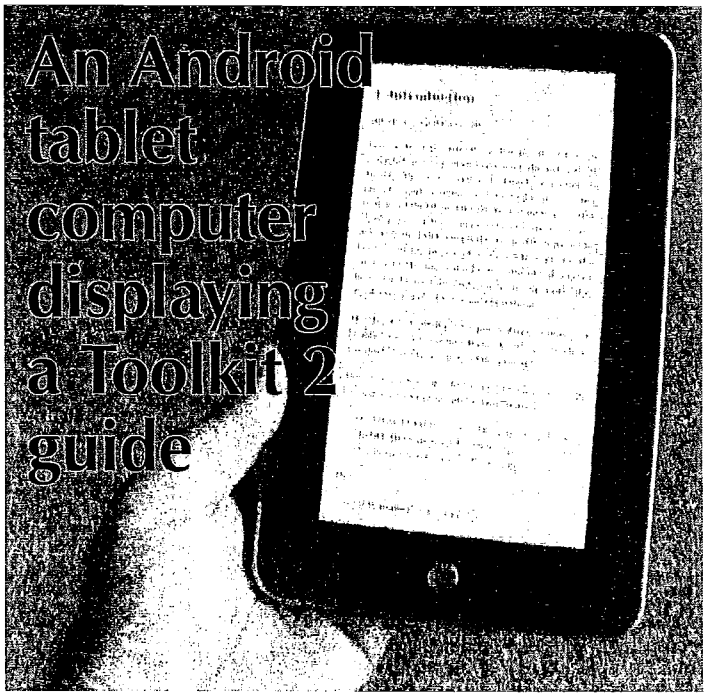
QL Today

Volume 16
Issue 4
June - August
2012

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

Read more about Dilwyn Jones' QL eBooks Initiative!



54 pages!

This is NOT a Win 8 magazine

Definitely Not!

The story of the SER-USB driver continues... Adrian Ives provides us with more details about his "serial Nightmare"

Time to Renew! If you have not done it yet, please use renewal form or renew online at www.QLToday.com

Ian Burkinshaw explains how to use the Parallel Port for other things than printing...

Did we mention **54** pages?

Another Games Review by Peter Scott

Just to be different!

Contents

Advertisers

in alphabetical order

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
email: smsq@j-m-s.com
email: QLToday@j-m-s.com

Editor:

Geoff Wicks
Flat 5b
Wordsworth Avenue
Derby DE24 9HQ
United Kingdom

Tel. +44 1332 271366
email: gtwicks@btinternet.com
email: QLToday@j-m-s.com

Co-Editor & UK Office:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 20 71930539
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL *Today* is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at http://www.dilwyn.uk6.net/arch/index.html

The deadline for the next issue is the 14th of August 2012!

Editorial

by Geoff Wicks

Recently there has been some discussion on the QL-users email group about an electronic version of Jan Jones' book, "QL SuperBasic - The Definitive Handbook".

The book was first published by McGraw Hill in 1985 and a few years later there was a limited reprint published by Quanta. For various legal and practical reasons Quanta could only sell it at cost price which was £8. Today you can buy a second hand copy of the Quanta reprint from Amazon for £40.

Jan Jones' book can no longer be called a definitive handbook. It was written long before the days of toolkits, the pointer environment, emulators and GD2 colours. Today's QL is very different from the QL in 1985, and, if anything, the present definitive work is RWAP software's SBASIC/SuperBasic Reference Manual. But it is a testimony to Jan Jones that her book is probably the most thumbed reference work among QL-ers.

In the last 12 months there has been a lot of emphasis on electronic versions of QL reference works for which Dilwyn Jones must take much of the credit. An electronic version of Jan Jones' book would be a valuable addition and Quanta is the best body to conduct any negotiations with her. They have had a good track record including paying substantial royalties on each copy of the reprint.

Quanta is currently in a difficult transition period, reported elsewhere in this issue, following the large rise in the subscription at the beginning of this year. However 120 people have shown their loyalty to Quanta by paying the increased subscription and that is no mean achievement after a quarter of a century. Especially now that it is the last remaining active national QL interest group. And should anyone doubt the relevance of Quanta today it should be remembered that the continued availability of keyboard membranes is down to Quanta.

Difficult times do not mean the end is necessarily nigh. About four years ago both QL Today and the Quanta Magazine were having serious problems that threatened their existence. Arguably both have emerged stronger from the crises. Unlike in the previous two years, this year we have had no hesitation in wanting to continue with a new volume of QL Today.

Volume 16 has been a success. (The statistics are for the last four issues, that is, issue 4 of volume 15 and issues 1, 2 and 3 of volume 16.) Our archive DVD was much appreciated and once again the magazine was thicker than promised. We guarantee 128 pages, but produced 160. Almost 84% of these were editorial, a slight reduction in comparison with the previous year. There has been slightly less news coverage and like the Quanta Magazine, with whom we co-operate on news stories, we increasingly have to search out the news. We have a team of 6 regular writers and during the year a further 7 have contributed. We would like to see more of both. The occasional contributors have an important role in keeping the magazine fresh and varied in content. One welcome trend in the last six months has been the amount of copy that is coming in well before the deadline date. Thanks to all the writers who have done that.

Thanks are also due to our loyal readers, some of whom have already renewed their subscription. Last year we had a slight rise in readership and one of the reasons we could run a thicker magazine than promised was the extra income from this. We hope you will remain with us and that next year we shall be looking forward to a volume 18.

QL Games for Windows

RWAP services have released a collection of commercial QL games to run on Windows PC's, using a specially written runtime version of QemuLator. The package of 10 games costs just £10. ***Rich Mellor*** says the release is intended to dispel the myth that the QL was solely a business computer, but he also hopes that it will widen interest in the QL.

"The idea behind the games pack (and the only way we could get the price so low, with the authors agreeing to substantially reduced royalties), was to make it have mass market appeal - i.e.. accessible to the wider PC market who may be interested to find out more about the QL and thereby to raise general interest in the QL itself."

He further adds that the decision to use QemuLator was because of the co-operation from the author, Daniele Terdina:

"Daniele has worked hard to produce a runtime version of Q-emuLator, which comes packaged with the games - there is no F1/F2 screen, no access to the original games files, it emulates the original QL speed, and people would not realise that the games are running on a QL emulator."

Rich Mellor added that he had considered using other emulators, but many games cannot be run on SMSQ-E and in any case QPC2 would be too fast. He already uses QLAY (or QL2K) when making QL adventures available for Windows, but that it is not suitable for arcade games because of the difficulty in altering the speed at which keys are read by the machine.

The package is not available for running directly on a QL system as the programs come in a single file containing the games and the emulator and cannot be extracted individually.

RWAP service can still supply the games individually in QL format, but have issued a caveat about a possible release as a package:

"Unfortunately, experience suggests that the majority of people who want to play games on the original QL have basically the standard black box, and no disk drives or disk interface (let alone extra memory as required for a couple of the games). I can still distribute games individually on microdrive, but they take a lot of time and effort to produce and then half the time, they no longer read when they are in the recipients hands, as the felt pad just disintegrates."

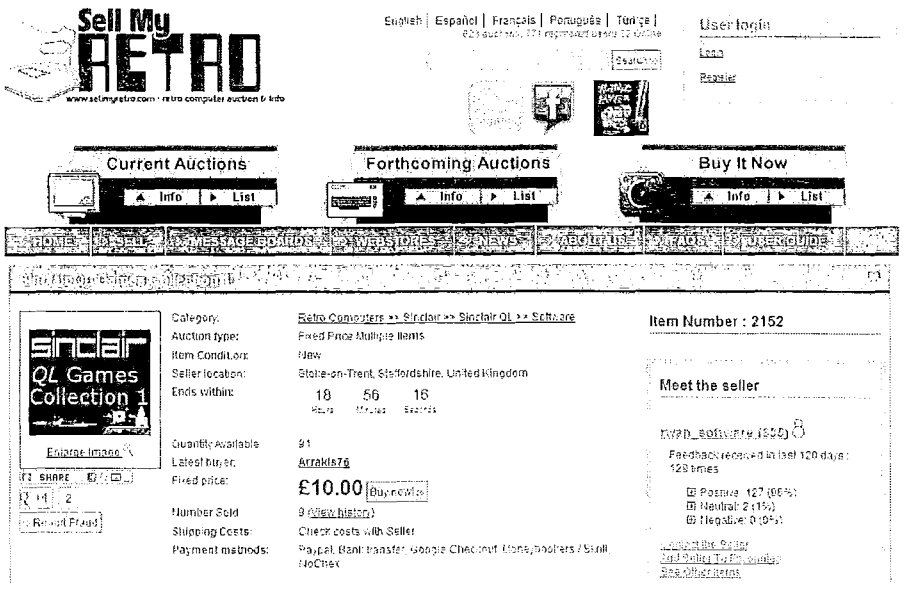
Peter Scott has uploaded a video of the games collection on YouTube:

http://www.youtube.com/watch?v=y4svJOT3Stw&feature=player_embedded

He has also written a review for this issue of QL Today.

The games package is available at:

http://www.sellmyretro.com/offer/details/Sinclair_QL_Games_Collection_1-2152



SMSQ/E Emulator for JAVA?

Shortly before the QL Today news deadline date ***Wolfgang Lenerz*** announced that he was developing an SMSQ/E emulator for Java 7, to be called SMSQulator.

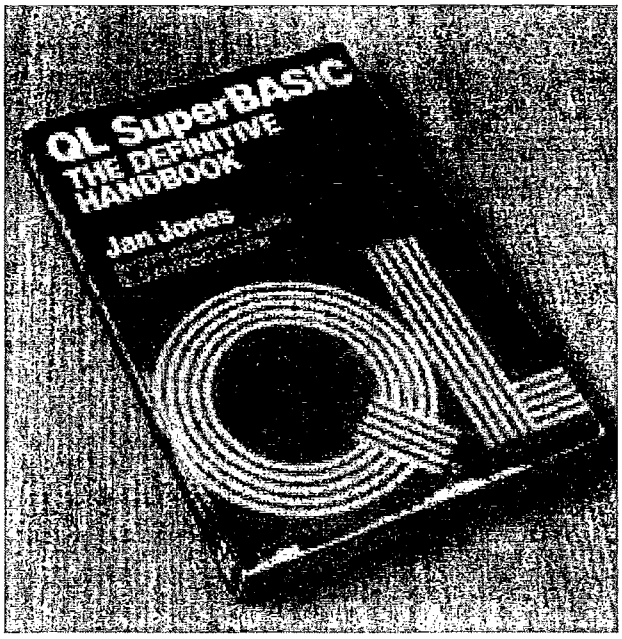
Initially Wolfgang described it as an adapted Gold Card version running in mode 4 on a 512 x 256 screen. There were problems with keyboard handling and there was only a Native File access drive (NFA). The emulation was also slow, probably about that of a native QL.

Following an appeal for technically skilled alpha testers he was able to report a fortnight later that progress was being made, and it was working under both Windows and Linux, but there were keyboard problems with a Mac. It was possible to

use a screen larger than 512 x 256 and basic, PE and WMAN were all working. Keyrow and mouse were also working and it was possible to move PE windows around the screen. However it was still running at less than an eighth of the speed of QPC2. There were also problems with some Qliberated programs, QMON, and C compiled programs.

SUPERBASIC E-BOOK?

Recently there has been discussion on the QL-users email group about the possibility of an e-book version of Jan Jones' "QL SuperBASIC" that many regard as being the definitive work on SuperBasic.



The book was first published in 1985 by McGraw-Hill who held the original copyright, but this reverted to Jan Jones in December 1987. In the late 1980s Quanta approached Jan Jones for permission to do a reprint of the book. It was with some reluctance that she agreed with a strict condition that her contact details were not to be revealed. Quanta also had some problems with distribution because of its status as a non-profit organisation for the benefit of its members. To avoid having to pay VAT on all its income Quanta could only sell the reprint at cost price and only to its members. The price was fixed at £8 of which £5 were royalties to Jan Jones and £3 the cost of production. Later Quanta made a second reprint, but this was by photocopy and was technically below standard.

The possibility of a further reprint or electronic publication is being put before the Quanta committee, but no early answer was anticipated because two of Quanta's officers were out of the country for an extended period.

Subscription Problems

Quanta is grappling with a problem of numerous members who have paid their 2012 subscription but not the full amount. When Quanta was founded in the early 1980s a popular way of paying subscriptions was by standing order in which members could ask their bank to pay a fixed amount to Quanta each year. Following the first rise in the subscription for about a quarter of a century many members have failed to inform their bank of the increase. Quanta is temporarily registering them as creditors and informing them that to continue their membership they need to make up the difference.

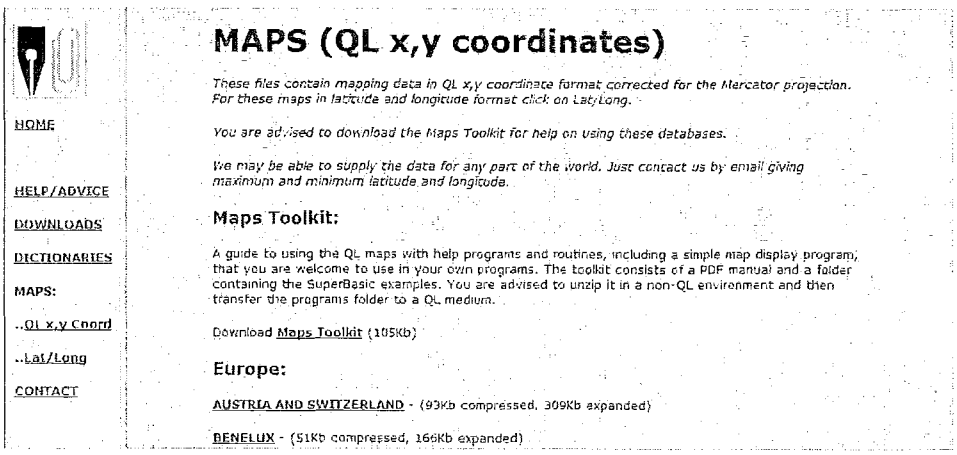
The problem was not unexpected. When a rise in the subscription was debated in the middle of the last decade, two officers argued against it precisely because they foresaw this problem arising. Almost three quarters of Quanta members have renewed their subscription at the new rates.

A full report of the Quanta AGM appears elsewhere in this issue.

Maps Web Page

Just Words! has now launched its QL friendly maps page. This is a series of databases containing outline maps in QL x,y format corrected for the Mercator Projection that QL-ers can use in their own programs. It is possible to print out whole countries or extract smaller areas from the databases.

A maps toolkit is available on the page containing several programs/routines including a simple



map display program that automatically calculates the optimum scaling for a map. It is also possible to reduce the size of a database; extract a small area from it; and convert a database into SuperBasic data lines.
www.gwicks.net/justwords.htm

Software Upgrades

QStripper

Norman Dunbar has updated his Qstripper program, which now can open PC as well as QL versions of Quill. He writes:
"I've updated QStripper and have finally got around to uploading the binaries as opposed to the plain old source code.
The changes are simple:

- *The help->About screen now displays a version number. The latest is version 1.01.*
- *The application can now open PC Quill as well as QL Quill documents.*
  Please note, however, until I get a definitive list, only some of the accented characters in the PC files will be translated. If anyone finds any that don't translate, let me know. I have managed to work out the PC versions of the following only:
  â ä à
  ç
  é ê ë è
  ï î
  ô ö
  ü û ù

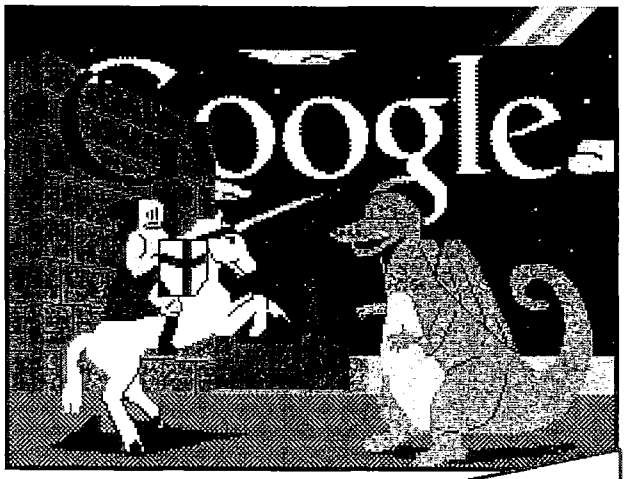
There are now versions for Windows 64 bit, Windows 32 bit and Linux 64 bit. At present I don't have a 32 bit Linux installation to build and test on. :-(
The download location is at
https://sourceforge.net/projects/qstripper/files/
and from there you can pick your required OS version.
In Linux, there are no support files required, just the executable. Under Windows, support files are required and there is a readme file in both Windows locations that tell you what you need. These support files are required by Windows only. You will only need them once until I start using QT 5, which isn't available yet!
Simple, but full, instructions are to be found in the assorted readme files at the download site."
Since the main news report was written Norman Dunbar has completed work on QStripper and launched a dedicated web site:
http://qstripper.sourceforge.net

TURBO

George Gwilt writes:
"Two faults in TURBO found by Michael Bulford (thanks) have been corrected in v5.07 which is available from my site."
http://gwiltprogs.info/

Google Honours Spectrum

St. George's day, the English patron saint, coincided this year with the 30th anniversary of the Spectrum. Google UK celebrated both on its UK home page.



News

Small Ads

FOR SALE

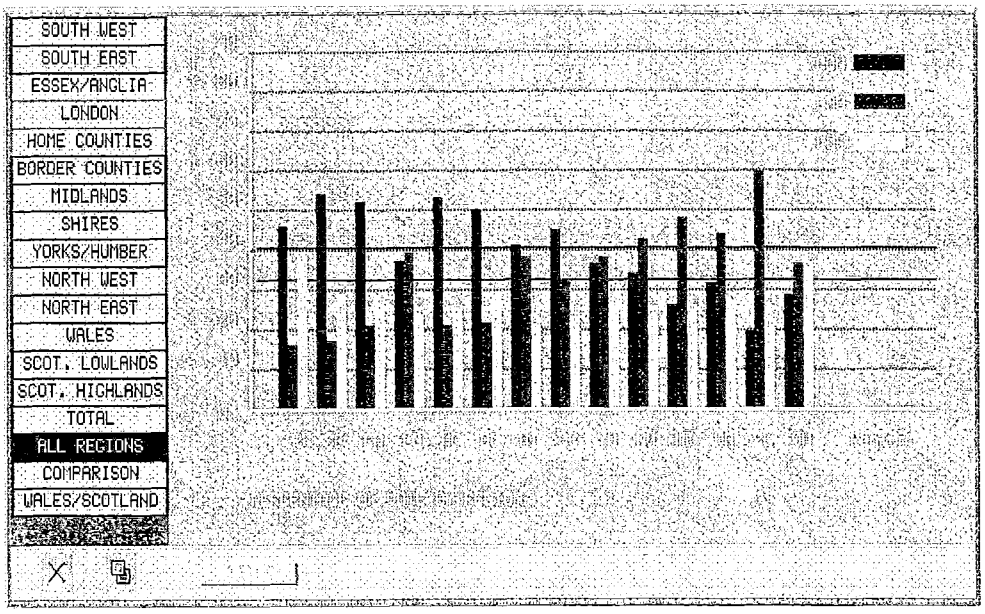
M68000 Family Reference book, 1988, approx 260 pages, gives full hardware data on M68000, M68008 and many peripherals.
M68020 Users Manual, 1989 approx 350 pages, contains details of capabilities, operation and programming including instruction set and bus timing.
Both by Motorola, £4 each including postage to UK only. John, **shortbutty@btinternet.com**

Maybe most of you forgot about it, but we still offer free private small ads to our subscribers. If you are searching for something, or you would like to sell or offer something, just send us a letter or an email with the text.
It should, of course, be QL-related, somehow...

CURSER or CURSOR?

by Geoff Wicks

In a previous article (QLT v16 i2 p18) we looked at the SCALE keyword and the differences between the way in which the QL handles text and graphics. No matter what the proportions of a screen in pixels the x,y ratio for graphics is always 1.37:1. When we program in graphics we have to think not in pixels but in graphics units, and the number of graphics units on a screen can vary according to the height of the SCALE command.



If we only programmed in graphics or only programmed in text we would have few problems, but more often than not we want to combine graphics and text on the same screen. This can be a problem best illustrated by a screen shot from one of my own programs (Fig. 1, to the left).

Strictly speaking I would not call this a true QL graphic as it uses the BLOCK command – even the lines are very thin blocks – and blocks are programmed in pixels for both placing and dimensions. However the graphic illustrates well the problems of trying to integrate graphics and text. The placing of a block can be adjusted by pixels, but text is placed by the AT command which, both horizontally and vertically, advances several pixels at a time.

Now lets go a stage further and imagine a true QL graphic on the same screen – that is it uses LINE, POINT, ARC, CIRCLE , ELLIPSE or Turtle Graphics. The screen is 600 x 300 pixels but 146 x 100 graphics units. To add to the complications pixels start at the top left hand corner of the screen but the origin of graphics units is fixed by the parameters of the SCALE command. You would have to do quite a few calculations to find the correct place to enter text and could soon find yourself using the sort of words, either under your breath or even out loud, that we are not supposed to use in QL Today.

Fortunately SuperBasic comes to our rescue with the CURSOR keyword that, in effect, does these calculations for us.

The syntax of the keyword is:

```
CURSOR [#channel,][graphics_position,]pixel_position
```

It is an unusual keyword because you use it in a different way in pixel screens from graphics screens. In her book "QL SuperBASIC" Jan Jones describes these as "simple" and "full".

SIMPLE is used when you are only working in pixels and has a simpler syntax:

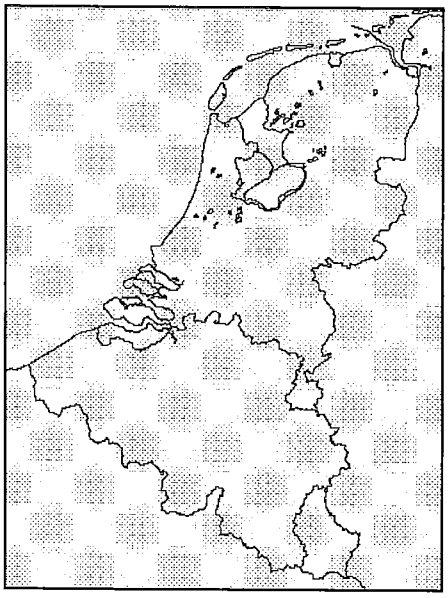
```
[line number] CURSOR[#channel,] pixel_position.
```

This simply moves the printing position to any desired place on the screen. If we look at the vertical axis on my graphic, which gives a percentage, then the coding is simple:

```
5685  CURSOR 30,195 : PRINT "0"
5690  FOR n=1 TO 9 : CURSOR 30,188-20*n : PRINT 10*n
```

The horizontal axis is a little more complicated:

```
5745 title$="SWSEEALOHCBCMISSYHNWNEWASLSH"
5750 r=1
5755 FOR n = 68 TO 380 STEP 24
5760  CURSOR n+4,210 : PRINT title$(2*r-1);title$(2*r)
5765  r=r+1
5770 END FOR n
```



If your graphic is a true graphic then you have to use the FULL syntax.

In this case you set the initial cursor position not by pixels, but by graphics units. You can then adjust the placing of the cursor using pixels. It may seem strange that you have to program the command in both graphics units and pixels, but the reason soon becomes clear with a practical example.

In Fig. 2 (to the right) there is a map of the Benelux countries. This is a true QL graphic drawn using the POINT command. We are now going to add the three capital cities, Amsterdam, Brussels and Luxembourg.

If my map reading is correct the bearings of these three cities are:

Amsterdam: 52.4N, 4.9E
Brussels: 50.9N, 4.2E
Luxembourg: 49.7N, 6.1E

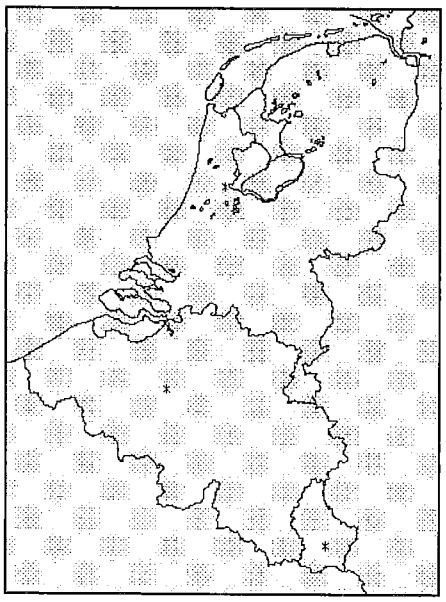
We have to convert these into QL x,y coordinates using the formulae:

```
x=.75*longitude
y=(180/PI)*LN(TAN((latitude*PI)/360+PI/4)
```

When you see these formulae don't fall into the error of thinking I am a brilliant mathematician and programmer. All the hard mathematical work was done by Hugh Rooms (QL Today v12 i3 p35).

Using these formulae we arrive at the following coordinates:

Amsterdam: 3.68,61.74
Brussels: 3.15,59.32
Luxembourg: 4.58,57.44



We can now add these to our map using the commands:

```
Amsterdam: CURSOR 3.68,61.74,0,0 : PRINT "*"
Brussels: CURSOR 3.15,59.32,0,0 : PRINT "*"
Luxembourg: CURSOR 4.58,57.44,0,0 : PRINT "*"
```

The 0,0 in each case means there is no displacement and the asterisk, a text character, that I have used to mark each city marks the exact location. (Amsterdam is difficult to spot, but it is there.)

When we want to add a name to each city it becomes a little more complicated. On our map Brussels is the easiest city because we have plenty of space for the name. We know the location is at x=3.15, y=59.32 and we need the name to be a little way away from this. We can work out roughly the displacement we shall need. QL characters are 10 pixels high and thus a vertical displacement of just over 10 pixels is needed. Brussels

has 8 characters which is equivalent to 48 pixels so a horizontal displacement of about 24 pixels could be used.

When we are printing characters it is much easier for us to think in pixels than in graphics units and that is the reason that the initial placing of the cursor is done in graphics units, but the displacement in pixels.

In practice this was the setting I used for Brussels:

```
CURSOR 3.15, 59.32, -20, 12 : PRINT "BRUSSELS"
```

Remember that when we are working in pixels the origin is at the top left hand corner of the window. This we have a negative value for the horizontal displacement.

Similarly for the other two cities:

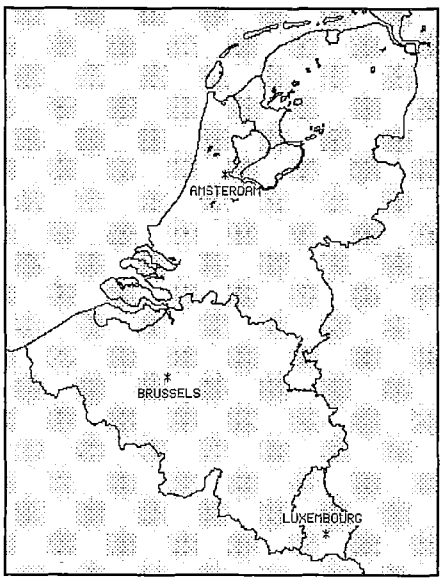
```
Amsterdam: CURSOR 3.68, 61.74, -24, 12 : PRINT "AMSTERDAM"
```

and

```
Luxembourg: CURSOR 4.58, 57.44, -30, -12 : PRINT "LUXEMBOURG"
```

In the last case I have printed the name above the city and not below and thus there is also a negative value for the vertical displacement.

CURSOR is a helpful keyword that saves us a lot of frustration and with a little thought is not difficult to use.



QL eBooks Initiative

by Dilwyn Jones

Over the Christmas period I heard from a number of people who'd recently got a Kindle eBook reader and it occurred to me that it might be possible to prepare some QL manuals and articles to be read on these devices. A busy few weeks followed as I set out to find out as much as I could about these devices and the file formats they supported and the result is the new QL eBooks section on my website.

These days, many QL users are returnees (used to have a QL, stopped using it years ago, now looking to restart using the QL as a hobby machine, or pure nostalgia) and new users who start using a QL as a retro machine or who have acquired one as part of a collection of older computers. So it makes sense to make QL documentation available in modern electronic formats. This helps avoid building up huge amounts of paper manuals, for example, not to mention the convenience of using modern electronic systems. After all, an eBook reader device can easily hold dozens if not hundreds of books in a device which is about the size of a slim paperback book, complete with screen - amazing how the march of technology has allowed both a battery and screen to be included in a device little more than a quarter of an inch thick!

I am very grateful to several people who helped me with valuable advice to help get my efforts going - people such as Norman Dunbar, Adrian Ives, Lee Privett and Bryan Horstmann to name but four - thanks everyone.

Having established that two file formats in particular (.ePub and .Mobi - which are in simple and vague terms zipped XML type files) allowed eBooks to be read on many different types of eBook platform such as Kindle, Android tablet and phone devices and even iPad. The Kindle uses a file format called ".azw" but it can also handle one of the above formats too, so luckily that was one less file format to worry about.

There are readers available for platforms such as Windows too, so it made it easy for me to test the files I created without having to buy a Kindle for myself. I already have an Android tablet computer which is good enough for eBooks, although its colour LCD display isn't quite as easy on the eye as a Kindle with its purpose made eInk display. Kindle screens are easier on batteries too than LCD screens! In fact, if you are thinking of doing some QL eBooks of your own, it is useful to have, say, the Kindle and Mobi

Pocket readers installed on your computer. They are not as comfortable on the eyes as a real e-reader device, but they are great for testing how your eBook will look as a finished product. Some of these readers have facilities such as change fonts used and change text size and quite a few other options, so you can experiment a little to see how well your eBook will stand up to use on different systems.

Get the Kindle for PC viewer free for PC, Mac, Android, Windows Phone, iPad and iPhone from Amazon at:

http://www.amazon.co.uk/gp/help/customer/display.html/ref=hp_left_ac?ie=UTF8&nodeId=200487800

Get the Mobipocket reader at:
http://www.mobipocket.com

One of the first things I was advised to do before trying to make an eBook was get hold of a program called Calibre (if you'd like to dabble in eBooks, get it from **http://calibre-ebook.com/** .) While this isn't available for the QL, it's very easy to use and can convert many formats of eBooks between each other. I started off converting PDF and Word .doc files but quickly found out that while this sort of worked, they weren't the best base file formats by any means. Calibre recommended working from HTML files – if you start off with a Word .doc file (as many of my QL documentation files are) you should use Word's facility to "Save As Filtered HTML" which gives the best results so far. Calibre lets you add such details as an eBook Cover (which can be as simple as a scanned copy of the original paper cover) plus author details, various descriptive search "tags" and a short text describing the book. It will not only convert HTML to the eBook formats mentioned above, it can convert from one format to another and it has quite a list of eBook formats it supports.

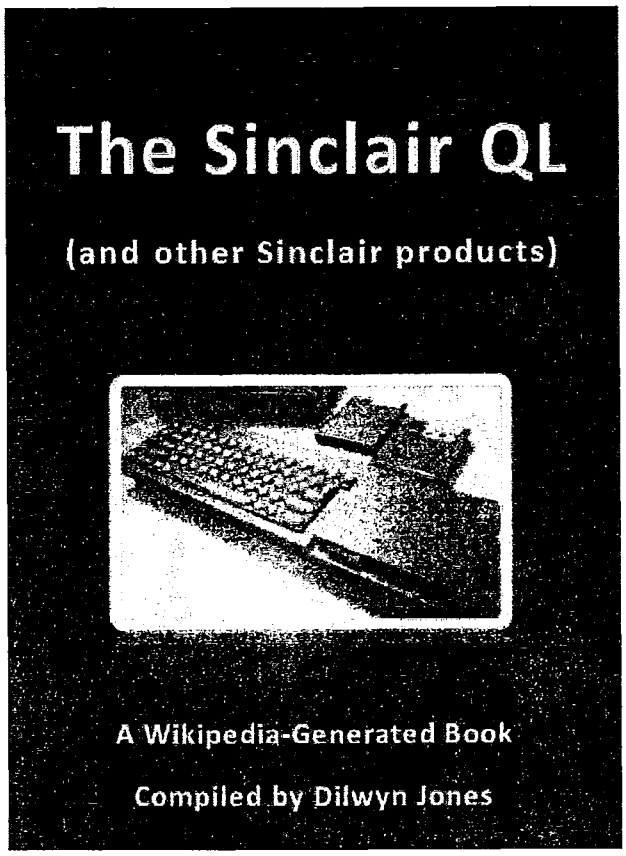


Figure 1: My Wikipedia QL book

Before I'd even got the project rolling to any degree (my first eBook effort was not worthy of the name, it was that bad), Adrian Ives of Memory Lane Computing sent me some eBook files he had created as reference guides for his own use with permission to add them to my website. So the ball started rolling with as yet little input from my part. Norman Dunbar also helped when I sent him a copy of his DJToolkit software manual as an eBook and he kindly came back with plenty of help and suggestions.

By now, a website was established at
http://www.dilwyn.me.uk/docs/ebooks/index.html

in the Documentation section of my website. I was aware that not all devices will download direct from a web page like this, but in most cases it was possible to download these eBook files on a PC or Mac or Linux system and transfer the files to the eBook reader either by wi-fi or a USB cable link. By dropping the eBook into the Kindle folder on my Android tablet computer the Kindle app found and added the eBook to its list of books. Some of these devices and reader programs will allow you to synchronise files across your various systems, making the eBooks available on all your devices capable of reading them.

At this stage I hadn't realised how easily it would also work with iPads, until Lee Privett mentioned it, He went as far as to create a video on YouTube showing how to download the QL eBooks to view on an iPad to display on the iPad's iBooks and Kindle viewers – his video can be seen at **http://www.youtube.com/watch?v=_0c_u1plqcU**.

So my web page started off with manuals such as the C68 guide, PDQC guide, Adrian Ives's The Shell manual and one or two others and many a late night followed preparing and editing manuals to add to the site. The list of what I wanted to add was getting very long, so it became a matter of prioritising and not tackling anything too big at that stage while I learned from my mistakes.

One of my biggest mistakes was relying on the use of the Tab key to space out columns of text – such columns became ragged and irregular. Norman Dunbar to the rescue, who patiently

explained that eBooks couldn't possibly be expected to know what every possible Tab spacing was and which one I intended to use. So, I could either use a monospaced font such as courier and neatly space out columns of text, or I could use simple tables to arrange the columns neatly. Consider for example a list of keywords and a brief one line description of them. My original efforts looked something like this:

PRINT-Write text to the screen
DEFine PROCedure-Define a procedure in SuperBASIC
BEEP-Make a simple bleep sound

So much nicer if they can be in neat columns like this:

PRINT	*Write text to the screen*
DEFine PROCedure	*Define a procedure in SuperBASIC*
BEEP	*Make a simple bleep sound*

So, what I did was to put such text into a two column table like this:

PRINT	Write text to the screen
DEFine PROCedure	Define a procedure in SuperBASIC
BEEP	Make a simple bleep sound

If anyone else is interested in making further eBooks, here is a tip I quickly learned. If using Word (or a word processor with a similar facility), create these columns of text separated by a TAB (or other character which doesn't occur in the text) and then use the Convert Text To Table command in the Table menu (INSERT ribbon in Word 2010), making sure that you indicate which separator you use between columns otherwise it won't know where to break the columns apart. I tend to use two column tables to keep things simple, but the above example could also have been a two column, three row table – most readers will cope with that, although it is always safer to keep things as simple as possible. Broadly speaking, as the Ebook readers render XML or HTML code, most fairly simple tables supported by a web page will probably work OK, but I tend to err on the side of caution and keep it as simple as I can. Of course, once you've got the layout how you want it to look you can then make the borders invisible by setting the border colour to invisible or 'no colour'.

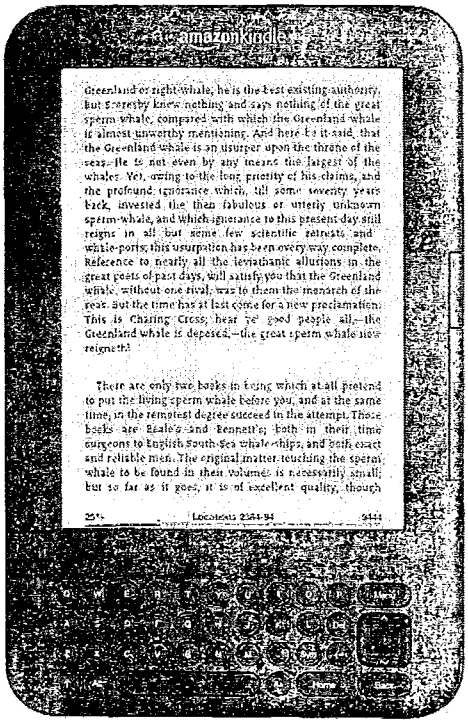


Figure 2: Amazon Kindle, version with

By now my confidence and experience were developing quickly, so I thought a major project would be a good idea to test my knowledge leaned so far. I happened to read an article in Computeractive magazine which showed how to use a Wikipedia facility to make a book out of selected Wikipedia pages – Wikipedia lets you do things like this as long as you quote them as the source of the information you use.

So I selected all the articles I could find about the QL, Sinclair and his other computers into one book which turned out to be a little over 250 pages long. How to write a 255 page book in about 20 minutes! I exported it from Wikipedia as a single book (very neat facilities they've got for doing things like this) and fed it into my word processor, straightened a few things out here and there, added an index (which I later found out I didn't need to) and set about converting it to eBook. Eek – about 30MB long! Well, OK, just treat that as a test of my eBook writing technique. Amazingly, it worked second time (first time I just forgot to give it a posh cover) and it took longer to upload the book to my website than it took to make it. ADSL upload is very slow compared to download speeds.

This book is available to download from my website's eBooks page at the address above, free of charge. Anyone who'd like a copy and has no broadband to download a file of this size is welcome to send me a CD and return postage to get hold of a copy. The book is called "The Sinclair QL (and other Sinclair products)"

By now the project was quickly gathering pace and Adrian Ives was kindly sending me QL-related eBooks almost as fast as I could add them to the website – quite a few of the eBooks on the site by early February were Adrian's work, thanks Adrian.

Then along came a major project – the QL manual. The thought of scanning and OCRing such a huge work frankly terrified me, knowing how much time I'd spent on this during January. Of course there was the plain text QL manual already on my website which I toyed with adapting and just as I was preparing to tackle that, Paulo Proietti wrote from Italy saying he had done just this very project, so I nearly bit his hand off accepting the offer and quickly added it to the website. The only slight issue with this particular eBook is that you have to view it using a fixed pitch font to be able to correctly see the diagrams, and the pictures from the QL manual aren't actually shown as pictures, they are ASCII character diagrams. I suppose they are good enough for a basic manual, but I still longed to be able to make a decent QL manual available. Former Quanta Editor Tony Hill had prepared a full QL manual, but that is available to Quanta members only (good reason to join Quanta just to get that!!!).

Then, another remarkable piece of luck occurred. Adrian Ives had, for his own use, OCRed a QL manual from a scanned copy by Andy Dansby of the World of Spectrum website, but it needed tidying up. So again a number of late night editing sessions followed until I had the Introduction, Beginner's Guide, Keywords and Concepts sections prepared and ready to go. So I converted the Word doc files to PDF, ePub and Mobi formats and added them to the web page as well. Then another idea struck me – I'd had to make a Filtered HTML copy to convert to eBook formats, so why not make it available as online HTML pages so that QL users who occasionally needed to refer to it (e.g. need to look up the syntax of a not-often used keyword) could just visit the website to view that part of the manual on an ad-hoc basis. With the Keyword Guide, for example, I added a list of keywords as links in the first couple of pages of the document, so that if you wanted to look up the TURNTO command you could just find it in the index, click on the link and it goes to the explanation for that particular keyword.

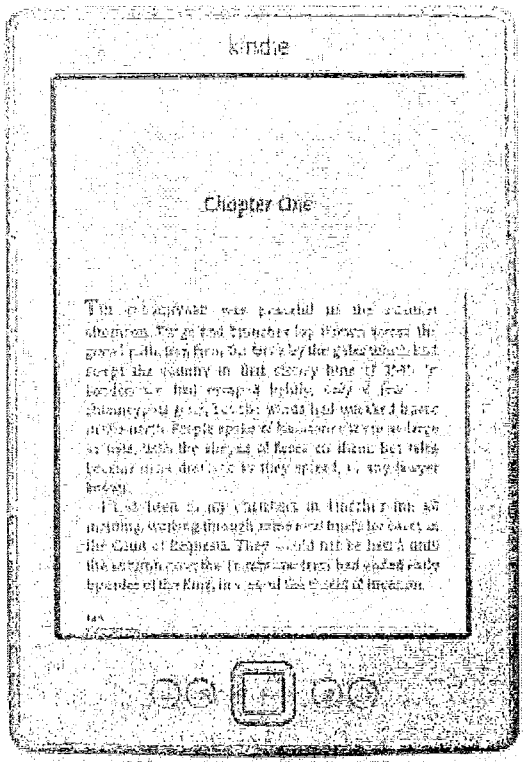


Figure 3: Kindle with touch screen, so no keyboard

So, if you need to look something up in the QL manual without downloading it or fetching the big heavy paper manual out just visit the web page at

http://www.dilwyn.me.uk/docs/ebooks/olqlug/index.htm

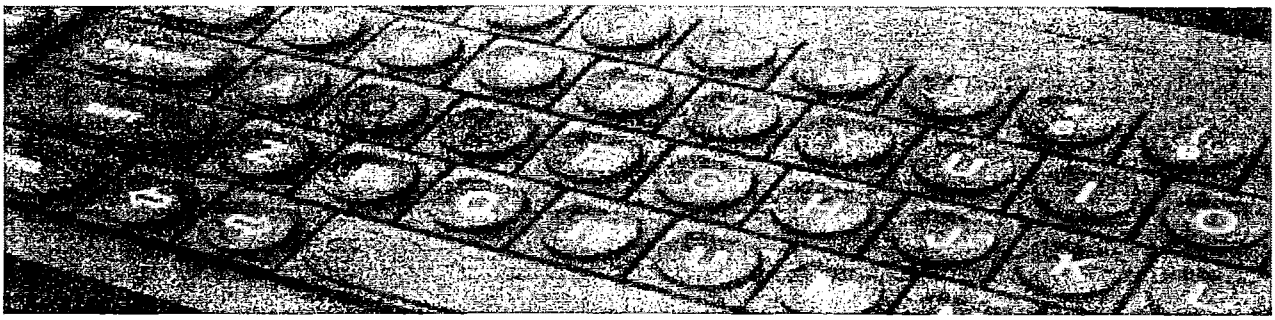
- at the time of writing it has the four sections mentioned above, and soon I hope to add the Toolkit 2 keywords guide (as most of us have Toolkit 2 in one form or another on our QL systems) and later I might tackle the Abacus, Archive, Easel and Quill manuals too,

A possible enhancement of the initiative in the future will be to work through the process of getting QL eBooks available through the Kindle market via Amazon. I haven't had a chance to look at this yet, but it seems like a worthwhile venture if time allows.

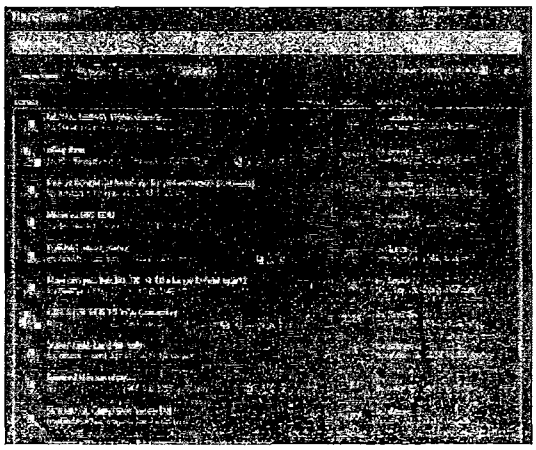
If you'd like to read more about eBooks and file formats, have a look at this article from Wikipedia:

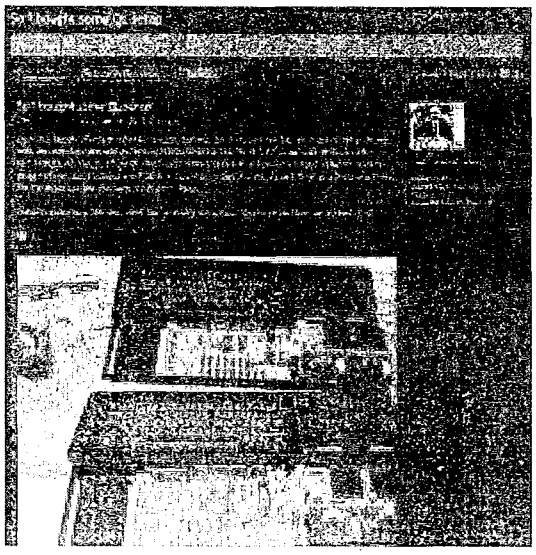
http://en.wikipedia.org/wiki/Comparison_of_e-book_formats

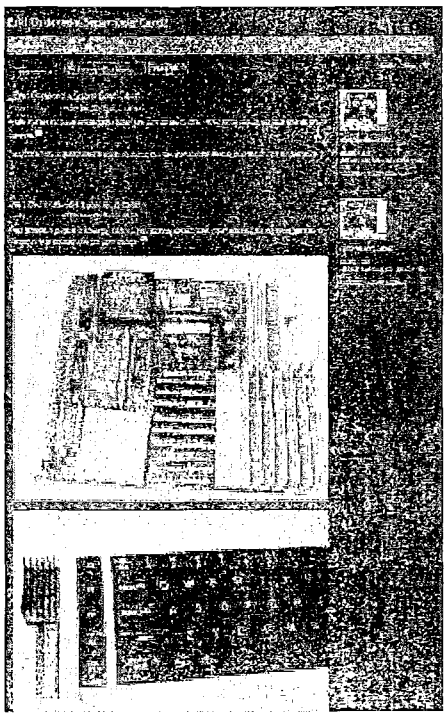
I apologise for making mention of so many non-QL subjects in this article, but I really hope that this initiative will help us all as QL users since many of us now have these devices since the prices became affordable. There must be times when we've all wished we had access to a convenient and portable copy of a document without having to keep a bookshelf or desktop full of paper! If you have created a QL-related eBook you'd like to make available, email me a copy to add to those already available from my website.



qlforum.co.uk







The online QL community. Free to join, no registration needed to view.

Please stop by for a visit!

A Serial Nightmare: The Story of the Ser-USB Drivers - Part 2

by Adrian Ives of Memory Lane Computing

Part Two: METAMORPHOSIS

I have every single revision of the Ser-USB driver; every milestone, every fix, every failed attempt to overcome the persistent serial I/O handling issues. There were 363 revisions of the driver code between January 11th, 2010 and October 4th, 2011, the day on which development of the 1.x series was formally brought to a close with the never completed v1.05, officially designated as 1.05.060-01. Over that time the driver's core was refined to become EDDE (Enhanced Device Driver Extensions) but the lion's share of the work was in trying to overcome the *D3 Curse*.

```
USB1
21076/21168 sectors
x_ref_txt

alloc_asm▒balloc, falloc, find_slot▒
atapi_asm▒chs_to_lba, read_lba, write_lba▒
basics_asm▒basic_procs▒
block_asm▒prepare_blockcall, r_blk0, r_pblk, r_pblk0, read_t
boot_asm▒boot_init▒
close_asm▒do_close▒
core_asm▒calc_csum, check_csum, dotidy_map, drv_inst, drv_lr
dbgout_asm▒dbg_out, dbg_out2w, dbg_outb, dbg_outl▒
delete_asm▒chk_empty, del_map, delete▒
extras_asm▒fil_ver, med_inf, med_xinf, q_date▒
file_asm▒in_load, out_save▒
flush_asm▒flush, flush_slaves▒
forced_asm▒do_forced, get_addr, get_slaved, getlblk, slaved▒
format_asm▒do_format▒
intserv_asm▒sched, driver_fifty_hz▒
io_asm▒▒do_IO, io_exit▒
iod_asm▒▒get_isector, iod_tbl▒
iorw_asm▒chk_pend, in_byte, in_line, in_pend, in_str, out_by
libcode_asm▒LongDivS, LongDivU, OutLong, q_str, RetInt, RetL
makedir_asm▒mk_dirs▒
open_asm▒complete, do_open, exit_open▒
posit_asm▒pos_abs, pos_rel, rd_head, set_head▒
rdsect_asm▒drive_read,drive_read_async▒
rename_asm▒rename, trunc▒
setup_asm▒base, hard_add▒
text_asm▒atapi_msg, bad_fat, bad_msg, blksz_msg, block_msg,
▒▒ide_msg, load_fat, mem_msg, mstr_err, noid_msg, qdos_err,
trashcan_asm▒dotrash▒
usbdev_asm▒boot, init▒
usbdrv_asm▒drive_capacity, drive_cmd, drive_nbusy, drive_sel
▒▒usbwiz_chkpipe, usbwiz_init, usbwiz_link, usbwiz_queue_adc
▒▒usbwiz_spawn_qm▒
util_asm▒dir_search, pre_conts▒
```

```
LRESPR 'USB_RAM_bin'
Serial USB Driver  v0.02β(2011)
Connected to a USBWiz v2.29 device on ser1hdr at 9600 baud
Loading and checking FAT ...
dir usb1_
view usb1_x_ref_txt
```

Illustration 3: Ser-USB driver 0.02 reading a file

Back then, in the early days, I knew about the D3 Curse and that, somehow, I would have to find a way around it. What I didn't know was that the implications of that problem would go deep into the very heart of the driver and would lead to another titanic struggle ... the Battle of the Slave Blocks. But that was still to come.

There was a long break in development between June 2010 and February 2011 during which no development took place. At that time we were relocating to Cornwall, so the Ser-USB project was mothballed. Development resumed on the 4th of February 2011 with a renewed determination to finish what had been started.

Here then are the highlights from the history of the driver's development from February to September 2011:

06-FEB-2011 v0.02.009

- Trap #2 io.formt is implemented. For the first time it is possible to issue the command FORMAT USB1_ from S*BASIC.

15-FEB-2011 v0.03.015 The Birth of the Queue Manager

This was the first version of the driver to incorporate the Queue Manager, my solution to the *D3 Curse*.

A mechanism was needed whereby it would be possible for the Ser-USB driver to make calls to the QDOS SER driver reliably. Zero timeouts (the recommended method) would not work because the call would never complete.

Previous attempts to get around the problem involved switching back into user mode and doing the trap with a defined timeout. This actually worked, but almost always resulted in a catastrophic system crash either immediately or at some time later, presumably caused by re-entering the scheduler whilst it was supposed to be inactive. SMSQ and

Minerva didn't seem to mind this, and didn't crash, but I wasn't prepared to create a driver that was limited to running under those environments.

It's worth mentioning that this problem can happen with any trap that re-enters the scheduler, particularly those that allocate and deallocate memory (for which there are vectored equivalents to be used from supervisor mode).

The Queue Manager implemented a simple queue mechanism which was monitored by a separate job. Transactions were placed into the queue, which were subsequently picked up and executed by the Queue Manager job. Because the QM was running in user mode, trap #3 calls to the serial driver worked fine.

In the driver, when a transaction was added to the queue, it returned immediately with err.nc (Not Complete) telling QDOS to retry the operation on the next scheduler loop. This became known as the "IOSS Retry Integration" feature of the driver. (It's actually the forerunner of the final solution adopted in the 2.0 driver, which uses retry integration, but without a Queue Manager).

Implementing the Queue Manager logic was relatively straightforward; the complexity came in handling the retries. It was necessary to suspend the driver's state before returning err.nc so that it could be restored when QDOS subsequently re-entered the trap on each retry. On the face of it, this would require some hugely complex conditional coding to navigate a re-entry path into the relevant part of the driver code and that meant rewriting most of the EDDE core, which would have taken a huge amount of effort. Actually, it would have meant starting again from scratch.

Instead, I adopted a system called the "Save State Engine". When a request was placed into the queue, the Save State Engine was called to save the driver's state. This involved not only saving all current registers, but also the stack right the way up to the top level.

On re-entering the trap servicer code, the Save State Engine restores the contents of the stack, pops the registers and resumes exactly where the driver left off.

The advantage of using the Save State Engine was that nothing had to be rewritten or even needed to have knowledge of what had happened.

The Save State Engine was, in fact, the hardest thing to write because it had to be transparent to the rest of the code. It was also the source of most of the problems with the whole Queue Manager solution.

And there is one huge flaw. It relies upon QDOS re-entering an incomplete trap until all of the serial I/O has been completed, but what happens if QDOS doesn't do that? A trap #3 call with a zero timeout (as I encountered later) and the doomsday scenario of "Out of Order Trap #3 Requests"; when more than one program is accessing the Ser-USB and they both issue trap #3 calls close together. The first call hasn't completed, so the Save State Engine is holding the driver's state, then along comes another request for a different transaction.

All of these problems would eventually be solved, but a rocky road lay ahead.

22-FEB-2011 v0.04.002

- Fixed the INPUT bug caused by improper slave block handling.
- Problematic initial handshaking under JM QDOS is fixed (yet again!) with a better retry and timeout mechanism.
- USB_RESTART command allows the driver to be restarted if it didn't detect the Ser-USB, or if you forgot to connect it.
- S*BASIC USB_PUTCMD, USB_GETCMD and USB_GETCMD$ interface to the driver layer is working.
- Implemented an auxiliary stack to get around QDOS's dreadfully small supervisor stack allocation when calling into the driver (even the original QUBIDE code had managed to exceed the 64 byte limit at one point).
- Driver is now layered to enable the hardware interface to be replaced.
- Default I/O mode is synchronous to preserve memory on low-end systems; asynchronous I/O functions are still available through the API for user programs.

Version 0.04 was a very important step in the driver's development. It marked the first version that I was prepared to release to public beta, but also the first version to begin the separation of the hardware and filing system layers that would ultimately result in the layered approach used by today's EDDE driver core.

One feature of this version survived right to the end of the 1.x series: the Auxiliary Stack. QDOS places a limit on the amount of space that a device driver can use on the supervisor

stack. Officially it's 64 bytes (which is utterly ludicrous) but, in practice, I found it to be safe to use at least twice as much. But 128 bytes is not a lot when you are saving and restoring contexts with multiple long word registers and nesting calls four or five levels deep.

The Auxiliary Stack used register a5 instead of a7 as a pointer to an area of heap allocated when the driver was started. All instructions to save context within the driver (move/movem instructions with register sets) used a5 instead of a7. This resulted in the supervisor stack only being used for return addresses and solved the problems associated with stack overflow. A5 had previously been used by the QUBIDE driver as a hardware base address pointer, so it was the obvious choice for this function, as Ser-USB had no need of such a pointer.

13-MAR-2011 Draft Ser-USB User Manual Released

To support the beta testers, the first version of the Ser-USB User Manual was released for public consumption. By this time the driver had advanced to version 0.07 and it was ready to be placed in the hands of a few brave testers.

14-MAR-2011 v0.07.004 First Public Beta Release

For the first time the driver was made available outside of Memory Lane Computing. From here on there was no going back. Completing the driver was now a matter of reputation.

I don't mind admitting that I had been dreading this moment. This is the moment when all of your mistakes are laid bare and there is nowhere left to hide.

28-MAR-2011 v0.08.018 Public Beta Release

- Fixed several issues, including RENAME not working and drive corruption if accessing a partition other than the first.
- Some commands were renamed:
  USB_DRIVE was now split into two commands:
  USB_MOUNT and USB_UMOUNT.
  USB_QM_START was now QM_START.
  USB_QM_STARTED() was now QM_STARTED()
- The driver could be unlinked from QDOS with the new USB_UNLOAD command, and the user could choose not to mount a drive as USB1 when loading.

At the time, I remember being quite proud of the USB_UNLOAD command; probably the first time that a driver could actually be unloaded safely. But these were also the early days of the Queue Manager, the user mode serial I/O sub-system that was intended to circumvent the *D3 Curse*, and there were still many problems ahead.

04-APR-2011 v0.90.003 Release Candidate 1

At this point the version number jumped to bring it closer to the hoped for 1.0 release. Beta testing had been going on for three weeks and I believed that the worst problems had been ironed out.

It hadn't been quite as bad as I had feared and I was optimistic that it would soon be possible to have a working product. I was wrong.

08-APR-2011 v0.91.001 Release Candidate 2

This release incorporated the remaining changes that were necessary to completely split the hardware layer off from the file system driver, meaning that the Ser-USB driver architecture now required only the replacement of a handful of hardware-specific functions in order to be re-used with different physical hardware. This was the first true incarnation of EDDE.

There were one or two other changes as well:

- One bug was fixed: A regression in RC1 that prevented USB_RESTART from working.
- One final piece of legacy code was dropped: drv.drive_capacity now only returned a value in LBAs (the deprecated CHS value was no longer supported, removing the last remnant of the QUBIDE architecture).
- Extensions and installable modules now used a link-loading system that reduced their size and allowed them to directly call much more of the core driver code (in particular, the S*BASIC support routines which were not originally exposed through the driver's API and so had to be duplicated in previous versions).
- The driver architecture was officially formalised around a new standard: EDDE (Enhanced Device Driver Extensions).
- Extension file names were changed to reflect the fact that they will work with any EDDE-compliant driver.
- Ser-USB Partition Manager and Ser-USB Status Monitor program names were changed (both programs would work with any EDDE-compliant driver).

- Some API functions had different names, but their function codes and vector numbers were unchanged.
- The names of most of the S*BASIC commands were changed from USB_xxx to DRIVER_xxx to reflect the fact that they were now generic and applicable to any EDDE driver.

Looking back, these were pretty big changes to have made just going from one release candidate to the next!

12-APR-2011 Decision to withdraw support for standard QLs

After reviewing a new tranche of reports from beta testing it became clear that my earlier optimism had been misplaced. There were big problems with the driver. It just didn't work on a standard QL without a Hermes or superHermes. It is worth reproducing below my posting to the QL Users List at the time. I think it explains everything, especially the frustration that I was feeling at the time:

> *It is with regret that I have to announce that I am withdrawing support for Ser-USB on QL hardware without superHermes (or better) enhanced serial ports.*
>
> *As of today the Queue Manager installable module, whose primary purpose was to support the base QL configuration, is also withdrawn. A final version had been developed which incorporated a completely new method of IOSS retry integration that seemed to be working extremely well. Much better, in fact, than any previous version - but once again the design of QDOS and the inherent unreliability of the standard serial ports prevented it from being useable. I have therefore decided that enough is - very definitely - enough!*
>
> *It has been a long and very costly process attempting to develop this driver, and with next to no interest in the device as a commercial proposition I can no longer devote resources to any further work. I will make Release Candidate 3 available later today. This will be the final version released to beta test. I will include the "last ever" version of the Queue Manager as a curiosity, if you wish to experiment, but you will find these issues:*
>
> *- You cannot copy executable files in Q-emuLator. This appears to be related to the way that the emulator intercepts fs.heads traps to save the header of a QDOS file on a DOS filesystem, but*
>
> *- ... executable files are corrupted when read back with the Queue Manager running on all configurations, but can be read fine without it. (Unfortunately that isn't much use, because you need the QM running in order to read the file in the first place on base QL hardware!)*
>
> *My plan is to make the public release of the (enhanced serial hardware only) driver available on the 1st of May, at which time the source code of the core driver will be released into the public domain.*
>
> *Many thanks for your interest and assistance.*

This was not, of course, the end of the Ser-USB project, but it was a serious "wobble moment" and it wouldn't be the last time that my exasperation with QDOS and those blasted serial ports would cause me to seriously consider abandoning it.

13-APR-2011 v0.91.004 Release Candidate 4

After receiving some feedback on the decision to withdraw support for base QL hardware I decided to take another look at the asynchronous I/O support to see if it could at least be made stable enough that it could be used with limitations. As a result, Release Candidate 4 was produced:

- The issue with executable files becoming corrupted when they were loaded from the Ser-USB through the Queue Manager appeared to be fixed. The most likely cause of this was an erroneously computed buffer pointer, although it remained a mystery why this did not present with other file types.
- Inability to write executable files under Q-emuLator; I had absolutely no idea why this was happening except that it was related to Trap #3 fs.heads which, under Q-emuLator, appeared to be called and then abandoned. (I have since discovered that this behaviour is introduced by Toolkit II which makes the call with a zero timeout and had absolutely nothing to do with Q-emuLator). Without the IOSS retrying that call, the result was that a hanging thread was left in the driver. This was a problem related to another issue: "out of order Trap #3 requests" that still hadn't resolved. I couldn't fix this so I did the next best

(worst?) thing. If the driver detected that this had happened then it wrote an explicit message to the console, tried to unwind the suspended thread, and returned error code "In Use". This unwinding was not always going to be successful but it was better than leaving the user unaware that a problem had occurred.

- The experimental (and completely unreliable) support to address this issue (bit 7 of the Async I/O Mode) was removed.
- Performance. I couldn't do anything about this. I couldn't do split baud rates (eg 19200 write/4800 read), because that would have entailed constantly switching the USBWiz Baud Rate between operations and that would have killed performance completely. So, without decent serial hardware, the Ser-USB was limited to 4800 baud on a standard QL and that was that. (This is still true today).
- The Queue Manager tasks were all renamed from USB_xxx to DRV_xxx.
- I/O Queue Request entries were retained in the queue for 10 ticks instead of 1; this made it more likely that an IOSS Retry into qm_do_async_io would correctly pick up a completed request instead of not finding it and so assuming that it had completed. The downside was that the queue would start filling up when there were high data volumes - something which I reasoned probably wouldn't happen much at 4800 baud!
- The Driver Status Monitor was updated to 1.03; it now correctly cleared its window if there were no queue entries to display, and displayed the return code next to the status of a queue entry.

At this point, despite my statement on the previous day, it was my intention to keep the Queue Manager and standard QL support in the driver package (with a very strong disclaimer). It was this decision that ultimately resulted in the Ser-USB becoming a product that people could actually buy.

The "wobble moment" had passed, but it was not to be the last.

14-APR-2011 Prototype PCBs ordered

An order was placed for PCBs to create a number of production prototypes. These would later become the Mark I Ser-USB and would be sold to early adopters.

17-APR-2011 Feedback from QUANTA and an offer of a Hermes chip

Rich Mellor e-mailed me with news from the recent Quanta meeting. It seemed that somebody was reading my development update postings to the QL Users list after all! He also obtained a Hermes chip which I subsequently purchased to support the ongoing development.

18-APR-2011 v0.91.006 Release Candidate 5

This release included what I believed to be a working fix for the issue of copying executable files. In fact, it included a fix for the whole "Out of Order Trap #3 Requests" issue, that I hoped would improve Queue Manager stability for all configurations. The driver now reported "IOSS Retry Abandoned" if it detected the problem but, in many cases, it would be possible to carry on processing.

This had been a horrendously complex issue to tackle and I will be honest in saying that, at that time, I was still no closer to understanding "why" the fs.heads problem existed at all. What was the logic in doing a filing system call with a zero timeout and then not bothering to check the return value? Whenever any fix is implemented without understanding why something was broken to start with there is obviously a risk that it will be making an already bad situation even worse but, under the circumstances, I decided that any fix was better than no fix. Especially as I really, really, wanted to bring the Ser-USB development to an end.

This is how the fix worked: If the driver received a Trap #3 request that is for the same channel ID as a previous (but still suspended - i.e abandoned) request, the driver resumed the previously suspended request for one scheduler time slice, then returned "Not Complete" but with the (unchanged) parameters of the new Trap #3 request. This told the IOSS to repeat the current request (timeout permitting) on the next slice of the scheduler's time. The whole process was repeated until the originally suspended request was finally completed, then, on the next scheduler slice, the new Trap #3 started to be serviced. The solution used only the documented mechanisms for I/O under QDOS so I had hopes that it would be reliable.

There were a few other very small changes under the hood to improve code organisation, and one small enhancement: MOUNT could now take a single parameter, so you could type MOUNT 1 instead of MOUNT 1,1,1 !

20-APR-2011 The Slave Block Argument

Frustrated by QDOS behaviour regarding the way in which it decides when the driver will flush slave blocks, I posted details to the QL Users list. The response from Laurence Reeves that "I believe you are misusing slave blocks." left me incandescent with rage. I definitely over-reacted to this at the time (sorry, Laurence) but that didn't get me any closer to solving the problem:

If your driver uses the slave block system (which the EDDE core, as derived from QUBIDE, did) then you are required to implement a service that flushes them to disk. This service is called by QDOS whenever it needs to free memory. I have no problem with this as a concept, but the way that it is implemented in QDOS is that your flushing code is not allowed to make an error return. It MUST flush the blocks and return with them written to disk. It cannot defer the operation.

This caused huge problems for the Ser-USB driver because that would require a lot of serial I/O, and serial I/O cannot complete while in supervisor mode! Up to that point slave blocks were flushed by creating a special class of transaction in the Queue Manager using its "Asynchronous Writes" feature. In short, this meant copying the slave block to a buffer, then returning to QDOS and writing that buffer asynchronously. This pointless process did actually work ... until genuine writes became interspersed with slave block writes and the whole system got tangled up. The result was a deadlock that ultimately led to my decision to remove slave block support and implement the Private Slave Block system that is now used by all EDDE drivers.

21-APR-2011 Decision to end the Ser-USB Project

After the months of hard work that I had put into the project, Laurence's comment had a (completely unintended) effect on me. I decided to give up and call it a day. I announced to the beta testers that I was bringing the Ser-USB project to an end.

This was the second, and definitely the worst, "wobble moment".

In the next issue you will be able to read how it all continued ...

QUANTA's uncertain Future

by Geoff Wicks

On paper Quanta membership has fallen by about a third since the subscription was raised at the beginning of the year, but in practice few people have actually ended their membership, and there is a solid core of 120 fully paid up members. More than enough to ensure the organisation's future.

The status of about a third of the members is uncertain. It is a penalty Quanta is paying for being around as long as the QL and for not raising its subscription for over 20 years. Way back in the 1980s a popular way of paying regular payments was by standing order. You asked your bank to pay Quanta £14 each year and then you could forget about it.

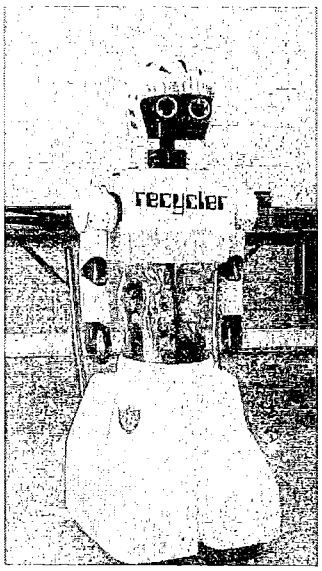
Unfortunately about 50 members have forgotten for a little too long and have not informed their bank of the subscription rise. They have paid a subscription, but not enough to continue their membership. Quanta has to contact each of these members asking them to make up the difference, and also suggesting that it is no longer a good way of paying their subscription, because there is now a postal supplement which is likely to rise each year.

Until this process is completed Quanta's future is uncertain. On paper subscription income has fallen by 16% and the part payers are being registered as creditors. Without a knowledge of how many will make up the difference and the repercussions for finance it is difficult for Quanta to prepare for the future, but no one is predicting its demise. This year's AGM was marked by positive thinking.

Attendance at the workshop was disappointing. It was eerily reminiscent of the last days of the Byfleet shows where you had tables around all four walls, but no one in the body of the hall. In practice the attendance was about the same as the last two years. There were 21 people at the show dinner and the AGM was easily quorate, although the attendance at presentations was lower than last year.

What appears to be happening is that shows are changing. There are fewer traders, but instead attendees are at their individual tables doing their own thing. This means there is much less circulation than there used to be.

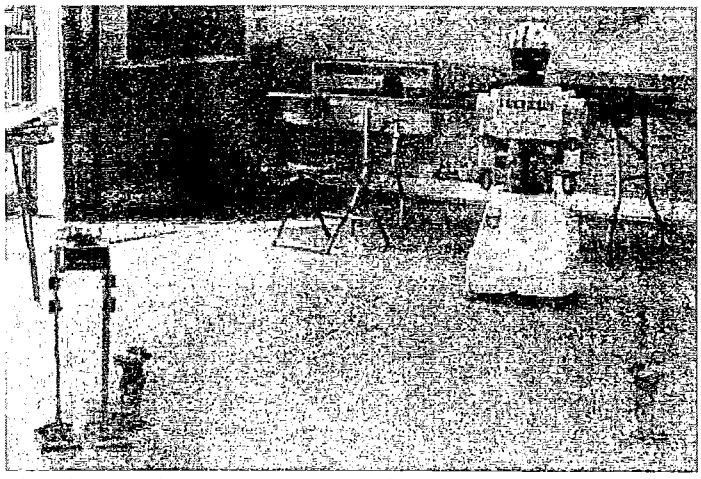
At a committee meeting immediately following the AGM, the committee briefly discussed the future of shows. Dilwyn Jones reported on his website: *"We also need to look at trying to find new venues for Quanta workshops and would consider applications from individuals as well as from subgroups who can offer a suitable venue. The fairly low attendance this year also means we need to look for ideas on how to try to encourage more QL users (not just members) to attend."*



The Chairman has had a makeover

The problem is that the traditional way in which Quanta has organised its workshops no longer works. In the past they were organised by a local subgroup, but now Manchester is the only subgroup with the resources, suitable premises, experience and desire to run a workshop. The Quanta committee have taken no initiatives in trying to revive interest in a show south of Birmingham.

One problem for Quanta is that UK shows have never had the social element that is/was present in most of the continental shows and the former North American shows, where a camaraderie compensates for the small number attending. (If you want to know what I mean by this go to **www.kuel.org** to discover what will happen at the Austrian show.)


The Chairman keeps a strict eye on her committee

The two presentations at the show - a talk on large numbers by George Gwilt and on Memory Lane Computing by Adrian Ives - had a lower than usual attendance, but were found interesting by those who were there. Adrian had a "double" show with a talk in the lecture room followed by a demonstration at his table of the QL-SD. Adrian reports that the main difficulties at the moment are the noise levels at the microdrive slot and some problems with Gold Cards. He thinks that the former is caused by the proximity of the microdrive slot to the video circuitry and he may have to recommend the use of the ROM slot. The latter he was unable to reproduce on a borrowed Gold Card at the show.

One innovation at this year's workshop was a wifi network. It was a fragile network because of a weak signal at the scout hut, but it did work. Early testing was done on the QL Today laptop situated at the far boundaries of the network. It seemed to be too weak a signal to be usable but in practice I was able to get a reliable internet connection.

In fact, if there was a theme at this year's AGM it was the electronic future of Quanta. One of the greatest financial headaches for Quanta over the next few years could be the Quanta Magazine, its largest single expenditure. Printing costs of the magazine take up somewhere between a half and three quarters of its income.

Editorially the magazine is becoming increasingly strong with an enthusiastic team of Lee Privett as editor and Dilwyn Jones as news editor. News now has a firm place in the magazine and Dilwyn has managed to completely revive the magazine's helpline, something that had eluded Quanta for about 15 years.

It was clear from discussion at the AGM that members see the long term future of the magazine as being electronic publication. Quanta has six or seven years experience of producing the magazine electronically, although uptake is still low. However in size and format the magazine is tailor made for use in e-readers and a massive rise in UK postal charges within days of the AGM will make it more attractive. Two suggestions made to increase interest in the electronic maga-

QUO VADIS DESIGN

Independant Information Technology Services

www.ql-qvd.com

QUO VADIS DESIGN Independant Information Technology Services

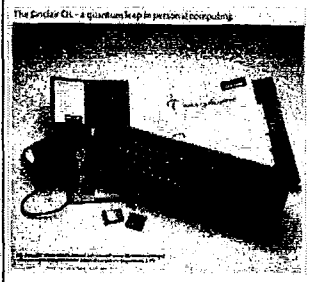
QL/QDOS/SMSQ/E Software

Home | Products | Support | Company | Contact

Welcome

Quo Vadis Design sells software for the Sinclair Quantum Leap computer (QL) and variants including a new O/S called SMSQE.

The QL is a computer in its 25th year Anniversary.



Software emulations of the QL now exist which can run on a PC/Mac with Windows/Linux or Mac Operating systems.

News

QVD QL News Blog - keep up to date
News Blog
24/02/2009

Quo Vadis Design Website Launched
01/02/2009

FEATURED PRODUCT

BUY NOW!

Copyright © 2009 Quo Vadis Design. All Rights Reserved.

Home | Products | Support | Company | Contact | Privacy

Special Offers available from Jochen Merz Software for its 25 years in QL Trading

Check the QL News Blog on our website for the special offers

Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

QL Today — The QL magazine for all QDOS, QL, SMSQ ... users!

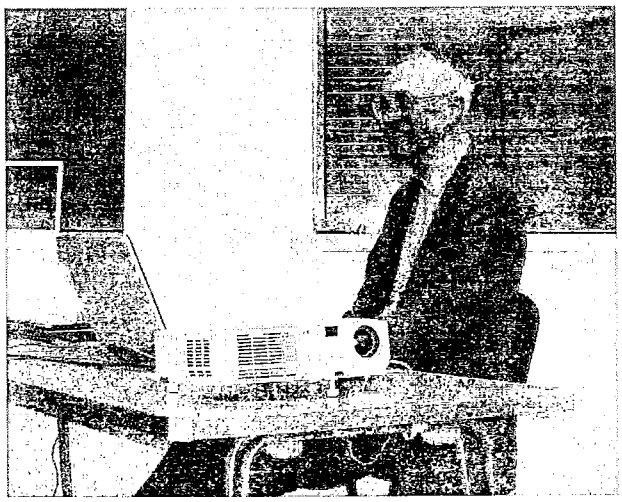
Subscriptions taken online

Running Quanta makes you hungry

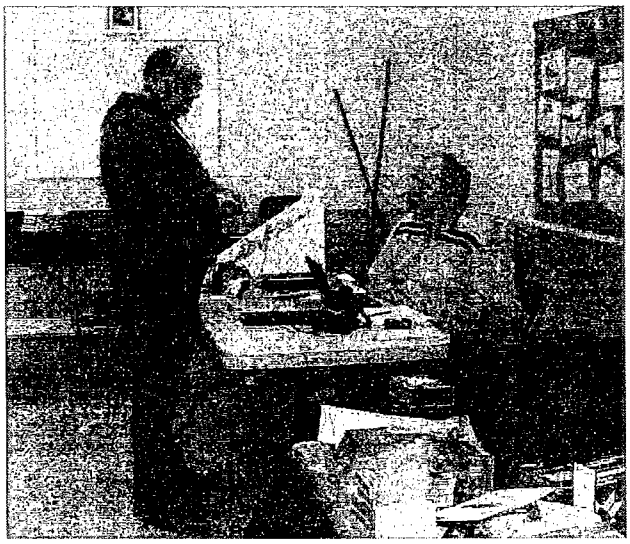
zine were a cover disk of back issues, and the placing on an e-book site for non-members to download for a fee.

Progress on developing Quanta's website has been disappointing or to use Quanta's own words "pretty static for the last 12 months". Keith Dunbar has now taken over full responsibility for the site and hopes to make more progress in the next 12 months, including implementation of the member's area. At the moment the news section of the site is fully operational and Quanta reports that several members are using the PayPal facility on the site to pay their subscription. Quanta does not have detailed statistics of the number and locations of hits, but I can report that 12 people have visited my website via the Quanta link.

Much Quanta committee business is now conducted by email although there was some doubt about whether this was constitutional. The AGM approved a substantially revised constitution to enable electronic participation in all meetings including the AGM. The new constitution is designed to give the committee more flexibility in its decision making, has simplified the wording of some clauses and removed ambiguities from others. These include the notorious clause 5.3 which took 47 words to say that the officers serve a three year period of office and did it without using the word "three".


"My number is bigger than your number"

Members voted to keep the three year term of office for officers and increase the term for ordinary committee members from one year to two. They also increased the maximum period a person can serve on the committee from six years to nine.


The Quanta Magazine is stunned by QL Today's captions

The last of these averted a crisis over the chairman as Sarah Gilpin was due to step down having served six years on the committee. She now can remain in office as does Alison Southern as secretary. Keith Dunbar has resigned as treasurer to take over full responsibility for the website following the resignation of Dan Abbott. John Gilpin has returned as treasurer, Dilwyn Jones remains as news editor and helpline coordinator; Lee Privett as Quanta Magazine editor; and Dave Buckley as librarian.

The next couple of years will not be easy for Quanta as it adjusts to a possible lower membership and income, but morale remains high. 120 people have shown that Quanta means something to them by paying the increased subscription and that is a solid basis for a strong organisation.

Assembler Discussions, continued

by George Gwilt and Norman Dunbar

Norman's *[ND]* answers to George's *[GG]* comments on Assembler – Part 30

[GG] Norman's article Part 30 on Assembler though short is both interesting and useful. Here are a couple of comments.

1. The use of "in win1_gwasl_libs_cls_in" to replace the two commands "IN" and "LIB" is a good idea, and one I had not thought of. Not only does it save typing but it also prevents lines inadvertently, and wrongly, being inserted between the commands on any alteration to the code.

2. I agree with Norman's remark on page 23 regarding equates. In my library I have one SYM_LST file which contains only a very small proportion of the symbols produced on assembly of the relevant source code. If I alter this source code and forget to edit out the unwanted lines in the new SYM_LST file I tend to get a large number of errors signalled when I INclude it in another assembler file. This will be because many of the symbols in the SYM_LST file are the same as ones in the assembler code in which the SYM_FILE is Included. It would indeed be useful to have a way of automatically excluding the unwanted EQUates. However, since the main use I make of SYM_LST files is to help me debug programs using QMON, I do not want to reduce the contents of a SYM file during assembly. Nor, for the same reason, do I want to position the exclusion in the program producing the SYM_LST file. This means that we need yet another program to trim a resulting SYM_LST file.

After reading Part 30, I devised a very simple way of doing the job. To indicate which labels are to be included in the amended SYM_LST file I determined that each label be marked in the original source code by an EQUate whose name is the required label headed by a short string, such as "LB_" and whose value is that of the relevant label.

Thus, in Norman's program I would add the lines

```
LB_CLEAR_SCREEN     EQU     CLEAR_SCREEN
LB_CLEAR_TOP        EQU     CLEAR_TOP
LB_CLEAR_BOTTOM     EQU     CLEAR_BOTTOM
LB_CLEAR_TO_EOL     EQU     CLEAR_TO_EOL
LB_CLEAR_LINE       EQU     CLEAR_LINE
```

After assembly there would appear in the SYM_LST file

```
LB_CLEAR_SCREEN     EQU     *+$00000000
LB_CLEAR_TOP        EQU     *+$00000004
LB_CLEAR_BOTTOM     EQU     *+$00000008
LB_CLEAR_TO_EOL     EQU     *+$0000000C
LB_CLEAR_LINE       EQU     *+$00000010
```

A simple S*BASIC program would copy these, and only these, lines from the complete SYM_LST file to a new file having stripped out the initial "LB_".

[ND] I think I have an easier way of extracting the library routine equates while missing the potentially common stuff, as the following example may hopefully show:

```
CLS_SCREEN          EQU     $00000020
CLS_TOP             EQU     $00000021
CLS_BOTTOM          EQU     $00000022
CLS_LINE            EQU     $00000023
CLS_END             EQU     $00000024
INFINITY            EQU     $FFFFFFFF
CLEAR_SCREEN        EQU     *+$00000000
JUST_DO_IT          EQU     *+$00000012
CLEAR_TOP           EQU     *+$00000004
CLEAR_BOTTOM        EQU     *+$00000008
CLEAR_TO_EOL        EQU     *+$0000000C
CLEAR_LINE          EQU     *+$00000010
```

I'm thinking that all the code routines themselves have a different format for their equate, being an offset from * while the others do not have such an offset.

As long as you want all the code routines in the library to be exposed then simply write that small SuperBasic (or indeed, assembler) program to search for and extract only those lines containing "EQU" and "*+$" perhaps?

I assume it's not possible for a SYM file to contain a negative offset?

[GG] It would seem unlikely.

[ND] Using this proposal, I'd expect to extract only the following:

```
CLEAR_SCREEN        EQU     *+$00000000
JUST_DO_IT          EQU     *+$00000012
CLEAR_TOP           EQU     *+$00000004
CLEAR_BOTTOM        EQU     *+$00000008
CLEAR_TO_EOL        EQU     *+$0000000C
CLEAR_LINE          EQU     *+$00000010
```

Which does expose JUST_DO_IT in addition to the others, I admit.

having said that, I like George's proposal to indicate which routines are to be exposed by EQUating them to a new label which has a common prefix to allow easy extraction. That way, you can easily hide the JUST_DO_IT routine that is exposed by my proposal.

I think I like George's idea better! ;-)

[GG] My problem was that the program I wanted included by LIB contained about 20 times as many labels (all headed by *+$ in the SYM_LST file) as the few which were needed. So I definitely did not want all the *+$ entries. So, not having at the time thought of any other means of extraction, I laboriously edited the SYM_LST file using QD.

Quite large Integers - Part 2

by George Gwilt

In a previous article I showed some assembler subroutines which would perform arithmetic on quite large integers. I discuss here how these could be incorporated in assembler code which could be CALLed by S*BASIC. This is a step towards practical use of the routines.

However, before embarking on this I should explain an addition made to the assembler routines already described. This amendment is intended to speed up division when the divisor is a power of 2, and this includes the number one.

Amendment to Base Routines

Just before the label divs4 seven instructions have been inserted as shown after move.l d1,d7. These use the extra code at divs8 or divs9 as required.

Insertion

```
        move.l    d1,d7          p2  (position of top bit)

        beq       divs9          divisor = 1
        movea.l   a1,a0          A0 -> divisor
        moveq     #0,d1          to get number of bits
        move.w    sze,d0         size of numbers
        bsr       gbit
        subq.w    #1,d1          1 bit? . .
        beq       divs8          . . yes - special action

divs4   movea.l   (sp),a0        A0 -> dividend
```

Extra Code

```
; Here the divisor is 2^f [f>0 and equals 32*w + b].
; The following code shifts the dividend down f bits if the divisor is
; less than the dividend. Otherwise the remainder equals the dividend
; and the quotient is zero.
; D0.L contains w | b where w is the complete number of long words in
; the shift and b is the remaining (0 to 31) bits.

divs8   movea.l   (sp),a0        A0 -> dividend
        move.w    sze,d0
        moveq     #-1,d1
        bsr       gbit           get position of 1st bit (p1)
        cmp.l     d1,d7
        bgt       divs_end1      quotient = 0 & remainder = dividend
        movea.l   (sp),a0        A0 -> dividend
        move.l    d7,d0          w | b (shift of w lwds & b bits)
        move.w    sze,d1         r
        subq.w    #1,d1          r - 1
```

```
          move.l    d0,d2
          swap      d2                  w
          sub.w     d2,d1               r - w - 1
          lsl.w     #2,d2               4*w lwd shift in bytes
          adda.w    d2,a2               adjust answer pointer
          moveq     #1,d6
          lsl.l     d0,d6
          subq.l    #1,d6
          move.l    d6,d2               remainder mask
          ror.l     d0,d6               top mask
          move.l    d6,d7
          not.l     d7                  bottom mask
          moveq     #0,d5               set previous top to zero
          bra       divs11
divs10    clr.l     (a0)+               clear lwd of remainder
divs11    move.l    (a0),d3             next long word
          ror.l     d0,d3               do bit shift
          move.l    d3,d4
          and.l     d6,d4               keep top bits for next long word
          and.l     d7,d3               keep bottom bits for this long word
          or.l      d5,d3               add in the top bits
          move.l    d3,(a2)+            insert answer
          move.l    d4,d5               prepare for the next long word
          dbf       d1,divs10
          and.l     d2,(a0)             adjust msw of remainder
          bra       divs_end1           exit

; Divisor is 1
; This copies the dividend to the quotient and sets the remainder to zero

divs9     movea.l   (sp),a0             A0 -> dividend
          movea.l   a2,a1               A1 -> quotient
          move.w    sze,d5
          lsl.w     #2,d5
          lea       (a0,d5.w),a0        -> end of number
          lea       (a1,d5.w),a1                "
          bsr       copy                copy dividend to quotient
          bsr       clrnm               clear remainder
          bra       divs_end1           exit
```

Now we can return to describing the use to be made of the assembler routines.

CALL Routines

One way of using CALL to access different routines in a piece of assembler code is to CALL the code at the various addresses of the different routines. Thus, one might perform addition by:

```
CALL asad+ad_address
```

and subtraction by:

```
CALL asad+su_address
```

where the relative addresses of the two routines in code loaded at asad are ad_address and su_address respectively.

If an amendment in the code changes the addresses of the routines the S*BASIC commands would have to be changed too. It would save the need to change the S*BASIC instructions if the assembler code could be CALLed always at the same address. Thus, my suggestion is that the different operations are made available not by altering the CALLed address, but by altering the first parameter, which is arranged to be the operation number.

This will make it easier to construct the S*BASIC program. But what of the arithmetic? How do we set and where do we store the integers? How long are they? What, in short, is the means of communication between the assembler code and S*BASIC.

I have set the assembler code to arrange for a specified number of stores to contain numbers each of a specified size expressed in long words. The assembler routines will, of course, set the integers as requested by S*BASIC. If there is an error, or if a question is asked, the relevant information is returned in a word at ans1, which is at byte 2 of the assembler code. In one case the complete answer is the decimal representation of a number. In that case the length of the number is given in the word at ans1 and the address of the string of digits is given in ans2, which is at byte 4 of the assembler code.

Operations

All operations are requested by up to five parameters in the CALL command, which is, generally,

```
CALL   address,pn,A,B,C
```

The operations allowed by the assembler code are these:

pn	*Name*	*Operation*
0	INITIAL	Set B spaces of A long words each
1	ADD	B + C -> A
2	SUBTRACT	B - C -> A
3	MULTIPLY	B * C -> A
4	DIVIDE	Quotient of B / C -> A
5	PUT	B -> A
6	COPY	B -> A
7	ZERO	A is cleared
8	NEGATE	A is negated
9	TEST	A is tested with -1, 0 or +1 put to ans1
10	DECIMAL	A is converted to a decimal string
11	COMPARE	ans1 is set 0 if A = B and 1 otherwise
12	ADJUST	B * 10^9 + C -> A
13	MOD	Remainder of B / C -> A
14	POWER	B ^ C -> A
15	SIZE	Sets ans1 to the number of long words in an integer
16	RANGE	Sets ans1 to the number of stores
17	COUNT	A - 1 -> A with result of TEST in ans1
18	COUNTB	A - B -> A with result of TEST in ans1

In most of these operations A, B and C are indicators of stores containing the numbers involved in the operation. Thus the first store is referred to as 0, the second 1 and so on. The exceptions to these values of A, B and C are the three following operations where the parameters may be the values required and not pointers to stores:

a) INITIAL where both A and B are the values to be set.
b) PUT where B is the number to be set in store A.
c) POWER where C is the integer power to which the integer in B is to be raised.

Initial

The operation INITIAL has to be activated before any other operation will work. It asks for space from the heap which is allocated as follows. First there is a number of bytes, rounded up to even, sufficient to hold a string of decimal digits representing the largest number which can be held in the number of long words requested as the size of integers. Then follow spaces for four numbers, called sp1 to sp4 and to be used as working space. The remainder of the space is earmarked for the user integers.

Format of Program

The program checks the parameters, performs the requested operation and returns to S*BASIC.

QUANTA

Independent QL Users Group

World-wide Membership is by subscription only,

Offering the following benefits: Bimonthly Magazine - up to 52 pages

Massive Software Library - All Free! : Free Helpline and Workshops

Regional Sub-Groups. One near you?

Advice on Software and Hardware problems

1 year Membership Subscription £18 (includes eMag)

If you want a printed copy of Quanta magazine, add the 2012 postage rates below

UK & NI £2.50, Europe £10.00, Rest of World £14.00

PayPal Surcharge about 5% - PayPal (see QUANTA Web Site)

Cash, Cheques and Postal Orders Accepted

*****Now in our Twenty Ninth Year*****

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane,Stretford
Manchester. M32 9EH(UK).
Tel. 0161 865 2872

mail: membership@quanta.org.uk

http://www.quanta.org.uk

Email: membership@quanta.org.uk and ask about our special 3 Year discount

Code

```
        bra     start

ans1    ds.w    1       error code or length of decimal string
ans2    ds.l    1       address of start of decimal string
sze     dc.w    0       number of long words in number
totn    dc.w    0       number of user number spaces
adr0    dc.l    0       address of alchp area
sp4     dc.l    0       address of sp4
sp3     ds.l    1           "       sp3
sp2     ds.l    1           "       sp2
sp1     ds.l    1           "       sp1
adr1    ds.l    1       address of 1st user number

progs   equ     18      highest operation number
nme_e   equ     sp4     address of the end of nme
sp4_e   equ     sp3             "           sp4
sp3_e   equ     sp2             "           sp3
sp2_e   equ     sp1             "           sp2
sp1_e   equ     adr1            "           sp1

; This macro sets up a table of relative pointers to the operations

kst     macro   number
k\@     set     0
prg     dc.w    p0-prg
        macl    here
k\@     set     k\@+1
        dc.w    p|k\@~-prg
        if      k\@‹\1
        goto here
        endif
        endm

        kst     progs

; The program code starts here.

start   cmp.w   #progs,d1
        bhi     bad_exit  ——›   Number too high
        move.w  sze,d0          Has INIT been done? . .
        bne     st3             . .  yes, OK for all progs
        tst.w   d1              Is it operation INIT? . .
        bne     bad_exit  ——›   . . no
        bra     p0              INIT

; Now test parameters
; The table tstp contains one byte for each operation. This byte
; determines which of the last three parameters are to tested for being
; in range.
; The value 7 indicates all
;           3 indicates the first two
;           1 indicates the first only
;           0 indicates none

tstp    dc.b    0,7,7,7,7,1,1,1,1,1,1,3,7,7,3,0,0,1,3

; At this point D1.L contains the operation number and D2.L - D4.L the
; values of the remaining parameters.

st3     lea     tstp,a0         test table
        movem.l d2-4,-(sp)      parameters to stack
        moveq   #2,d5           count of 3
        move.b  (a0,d1.w),d0
st1     move.l  (sp)+,d6        next parameter (D4 to D2)
        lsr.b   #1,d0           test next bit
        bcc     st2             no test needed for this parameter
        sub.w   totn,d6         is pointer in  range? . .
        blt     st2             . . yes
        bset    #31,d5          mark error
```

```
st2       dbf       d5,st1              count three parameters
          tst.l     d5                  error found? . .
          bmi       bad_exit  ----->   . . yes

; Get program

          add.w     d1,d1               2*operation number
          lea       prg,a0              jump table address
          adda.w    (a0,d1.w),a0        add relative pointer
          jmp       (a0)                jump to operation

bad_exit  moveq     #-15,d0             bad parameter
          rts                           return to S*BASIC


; Supporting the programs below are six subroutines listed at the end.

; 1. set_ad sets number pointers to addresses relative to adr1 set in A3.L
; 2. getsp gets space from the heap and . .
; 3. relsp returns it
; 4. set_sgn codes the signs of B and C before a multiply or divide and . .
; 5. rst_sgn uses this information after the operation
; 6. LB4 finds the decimal digits in a number

; When an error occurs the return to S*BASIC is via bad_exit1 which sets
; -15 in ans1.


******************
* INITALISE (p0) *
******************
lg1       dc.w      30103               100000*log2
lg2       dc.w      3125                100000/32

; p0 sets sze, gets heap space and sets space pointers
; D2 = sze : D3 = n (number of user number spaces)
;
; If sze = 0, its stored value is cleared and space is returned to the heap.

p0        move.w    sze,d1
          tst.w     d2
          bne       p0_4                to set new space
          tst.w     d1                  old sze? . .
          bne       relsp               clear and finish
          moveq     #0,d0
          rts
p0_4      tst.w     d3                  any spaces requested? . .
          beq       bad_exit ----->     . . no
          tst.w     d1                  is there space allocated now? . .
          beq       p0_5                . . no
          bsr       relsp               return space
p0_5
          lea       sze,a3
          move.w    d2,(a3)+            set sze
          move.w    d3,(a3)+            totn = n
          move.w    d2,d0               sze -> D0.W
          mulu.w    lg1,d0
          divu.w    lg2,d0
          addq.w    #1,d0
          move.w    d0,d7               no of decimal digits needed
          addq.w    #1,d7
          bclr      #0,d7               make even
          ext.l     d7
          moveq     #4,d1
          add.w     d3,d1               n + 4
          mulu.w    d2,d1               (n + 4)*sze
          lsl.l     #2,d1               no of bytes
          add.l     d7,d1               add the name space
          bsr       getsp               get the space needed
          beq       p0_1                OK
p0_3      lea       sze,a0
```

```
          clr.w     (a0)                mark no space
          bra       bad_exit  ---->
p0_1      move.l    a0,(a3)+            set adr0 (address of ALCHPd space)
          adda.l    d7,a0               D7.L -> end of name space
          lsl.w     #2,d2               bytes / integer
          moveq     #4,d0               for 4 spares (sp1 to sp4) and adr1
p0_2
          move.l    a0,(a3)+            set addresses (sp1 to sp4 & adr1)
          adda.w    d2,a0               to next address
          dbf       d0,p0_2
          moveq     #0,d0
          rts                           exit

*************
*  ADD (p1) *
*************
; p1 A = B + C
; D2 -> A : D3 -> B : D4 -> C
; NOTE any of A, B and C may be equal to one of the others.

p1        lea       addnm,a5
p1_1      bsr       set_ad

; Copy B to sp4
          lea       (a3,d3.w),a0        A0 -> B
          lea       (a0,d5.w),a0        A0 -> B end
          movea.l   sp4_e,a1
          bsr       copy

; C + or - sp4 to sp4
          lea       (a3,d4.w),a0        A0 -> C
          lea       (a0,d5.w),a0        A0 -> C end
          movea.l   sp4_e,a1
          jsr       (a5)                add or subtract
          bne       bad_exit1 ---->

; Copy sp4 to A
          lea       (a3,d2.w),a1        A1 -> A
          lea       (a1,d5.w),a1        A1 -> A end
          movea.l   sp4_e,a0
p1_2      bsr       copy
          bra       pend                exit setting ans1

*****************
* SUBTRACT (p2) *
*****************
; p2 C - B -> A
; D2 -> A : D3 -> B : D4 -> C
; NOTE any of A, B and C may be equal to one of the others.

p2        lea       subnm,a5
          bra       p1_1

*****************
* MULTIPLY (p3) *
*****************
; p3 B * C -> A
; D2 -> A : D3 -> B : D4 -> C

p3        bsr       set_ad
          bsr       set_sgn             ABS(B and C) and mark D5.TOP

; Clear sp4
          movea.l   sp4,a0
          bsr       clrnm

; B * C -> sp4
          lea       (a3,d3.l),a0        A0 -> A
          lea       (a3,d4.l),a1        A1 -> B
          movea.l   sp4,a2
          bsr       muls                B * C to sp4
```

```
          move      ccr,-(sp)          keep error code
          bsr       rst_sgn            reset B and C
          move      (sp)+,ccr          restore error code
          bne       bad_exit1 ---->

; sp4 -> A
          lea       (a3,d2.w),a1       A1 -> A
          lea       (a1,d5.w),a1       A1 -> A end
          btst      #17,d5             negate? . .
          beq       p3_1               . . no
          movea.l   sp4,a0
          bsr       neg
p3_1      movea.l   sp4_e,a0           A0 -> end of sp4
          bra       p1_2

*************
* MOD (p13) *
*************
; p13 MOD B/C to A
; D2 -> A : D3 -> B : D4 -> C

p13       moveq     #-1,d6             mark MOD
          bra       p4_1

***************
* DIVIDE (p4) *
***************
; p4 B/C -> A
; D2 -> A : D3 -> B : D4 -> C

p4        moveq     #0,d6              mark DIV
p4_1
          bsr       set_ad
          lea       (a3,d4.w),a0       A0 -> C
          bsr       tstnm
          beq       bad_exit1 ---->    C = 0
          cmp.w     d3,d4              B = C? . .
          beq       p4_5               . . yes
          bsr       set_sgn            ABS(B and C) and mark D5.TOP

; copy B to sp4
          movea.l   sp4_e,a1           A1 -> sp4 end
          lea       (a3,d3.w),a0       A0 -> B
          lea       (a0,d5.w),a0       A0 -> B end
          bsr       copy               B -> sp4

; clear sp3
          movea.l   sp3,a0
          bsr       clrnm

; sp4/C to  sp3 with remainder to sp4
          movea.l   sp4,a0
          movea.l   sp3,a2
          lea       (a3,d4.w),a1
          bsr       divs
          bsr       rst_sgn            reset B and C

; copy sp3 to A (quotient)   DIVIDE  (D6 = 0)
; copy sp4 to A (remainder)  MOD     (D6 = -1)

p4_6      lea       (a3,d2.w),a1       A1 -> A
          lea       (a1,d5.w),a1       A1 -> A end
          tst.l     d6
          bmi       p4_2               MOD
          btst      #17,d5             negate? . .
          beq       p4_3               . . no
          movea.l   sp3,a0
          bsr       neg
p4_3      movea.l   sp3_e,a0
          bra       p1_2
```

```
; MOD
p4_2      btst      #16,d5              negate? . .
          beq       p4_4                . . no
          movea.l   sp4,a0
          bsr       neg
p4_4      movea.l   sp4_e,a0
          bra       p1_2

; B = C so DIV = 1 and MOD = 0
p4_5      movea.l   sp4,a0
          bsr       clrnm               MOD -> 0
          movea.l   sp3,a0
          bsr       clrnm               DIV . .
          move.w    #1,-2(a0)           . . -> 1
          bclr      #16,d5
          bclr      #17,d5
          bra       p4_6                set answers

*******************
* PUT NUMBER (p5) *
*******************
; p5 N put in A
; D2 -> A : D3, -> N (number)

p5        move.w    sze,d5
          lsl.w     #2,d5
          mulu.w    d5,d2               relative address
          movea.l   adr1,a3             base address
          lea       (a3,d2.w),a0        A0 -> A
          bsr       clrnm               sets A0 -> end
          move.l    d3,-(a0)            enter number
          moveq     #0,d0
          rts

********************
* COPY A to B (p6) *
********************
; p6 copies A to B
; D2 -> A : D3 -> B

p6        bsr       set_ad
          lea       (a3,d2.w),a0
          lea       (a0,d5.w),a0        A0 -> A end
          lea       (a3,d3.w),a1
          lea       (a1,d5.w),a1        A1 -> B end
          bra       copy

*************
* ZERO (p7) *
*************
; p7 zeroes A
; D2 -> A

p7        bsr       set_ad
          lea       (a3,d2.w),a0        A0 -> A
          move.w    sze,d0
          bra       p7_1
p7_2      clr.l     (a0)+
p7_1      dbf       d0,p7_2
          moveq     #0,d0
          rts

***************
* NEGATE (p8) *
***************
; p8 negates A
; D2 -> A

p8        bsr       set_ad
          lea       (a3,d2.w),a0        A0 -> A
          bsr       neg
          bra       pend
```

```
*************
* TEST (p9) *
*************
; p9 tests A
; The result (-1 ,0, 1 or -2) is put in the word at ANS1.
; D2 -> A

; The result is;
;  1 = positive non zero
;  0 = zero
; -1 = negative (can be negated)
; -2 = largest negative (can't be negated)

p9        bsr       set_ad
          lea       (a3,d2.w),a0        A0 -> A
p9_2      moveq     #-2,d2              provisional answer
          bsr       tstnm
          bmi       p9_1
          bne       p9_e1               +1
          bra       p9_e2                0
p9_1      tst.w     d0
          bmi       p9_e4               -2
          bra       p9_e3               -1
p9_e1     addq.w    #1,d2
p9_e2     addq.w    #1,d2
p9_e3     addq.w    #1,d2
p9_e4     lea       ans1,a0
          move.w    d2,(a0)
          moveq     #0,d0
          rts

*******************
* TO DECIMAL (p10) *
*******************
; p10 sets a number to a decimal string
; The length of the string is set in ans1
; The address of the start of the number is in ans2
; D2 -> A

p10       bsr       set_ad
          lea       (a3,d2.w),a0        A0 -> A
          lea       (a0,d5.w),a0        A0 -> A end
          movea.l   sp4_e,a1
          bsr       copy                A -> sp4

; Set D3 and negate if needed
          moveq     #0,d3               provisionally +
          movea.l   sp4,a0
          bsr       tstnm               test the number
          beq       p10_2               print zero
          bpl       p10_1               positive non zero
          moveq     #-1,d3              set negative marker
          tst.w     d0
          bmi       p10_1               don't negate
          movea.l   sp4,a0
          bsr       neg                 negate
p10_1     movea.l   sp4,a0
          bsr       lb4                 set decimal string
p10_3     lea       ans1,a0
          move.l    nme_e,d0
          sub.l     a2,d0
          move.w    d0,(a0)+            . . to ans1 and . .
          move.l    a2,(a0)             . . string address to ans2
          moveq     #0,d0
          rts

p10_2     movea.l   nme_e,a2
          move.b    #'0',-(a2)          zero digit
          bra       p10_3
```

```
***********************
* ADJUST NUMBER (p12) *
***********************
; p12 A = B * 10^9 + C
; D2 -> A : D3 -> B : D4 -> C

ten9      dc.l      $3b9aca00          10^9

p12       bsr       set_ad

; Set 10^9 in sp3
          movea.l   sp3,a0
          bsr       clrnm
          move.l    ten9,-4(a0)

; Put B*10^9 in sp4
          movea.l   sp3,a0             A0 -> 10^9
          lea       (a3,d3.w),a1       A1 -> B
          movea.l   sp4,a2             A2 -> sp4
          bsr       muls

; Add C to sp4
          lea       (a3,d4.w),a0       A0 -> C
          lea       (a0,d5.w),a0       A0 -> C end
          movea.l   sp4_e,a1           A1 -> sp4 end
          bsr       addnm
          bne       bad_exit1 ---->

; Put this to A
          movea.l   sp4_e,a0           A0 -> sp4 end
          lea       (a3,d2.w),a1       A1 -> A
          lea       (a1,d5.w),a1       A1 -> A end
          bra       copy               Set answer and return

*****************
* COMPARE (p11) *
*****************
; p11 tests whether A = B or not
; D2 -> A : D3 -> B

p11       bsr       set_ad
          lea       (a3,d2.w),a0       A0 -> A
          lea       (a3,d3.w),a1       A1 -> B
          moveq     #0,d7              provisionally equal
          bsr       comp
          beq       p11_1              Equal
          moveq     #1,d7              Mark not equal
p11_1     lea       ans1,a0
          move.w    d7,(a0)
          moveq     #0,d0
          rts

***************
* POWER (p14) *
***************
; p14 sets B^C to A (C is the explicit power)
; D2 -> A : D3 -> B : D4 -> C

p14       move.l    d4,d6              Keep C and clear D6.TOP
          bmi       bad_exit1 ---->
          bsr       set_ad
          lea       (a3,d3.w),a0       A0 -> B
          bsr       tstnm
          beq       bad_exit1 ---->    B must not be zero
          bpl       p14_6              +
          lea       (a3,d3.w),a0       A0 -> B
          bsr       neg
          bset      #16,d6             mark to negate B at end
          btst      #0,d6              odd power? . .
          beq       p14_6              . . no
          bset      #17,d6             mark answer to be negated
```

IF YOU HAVE NOT RENEWED YET FOR QL TODAY VOL. 17, PLEASE DO SO AS SOON AS POSSIBLE! THE BETTER WE CAN PLAN AHEAD THE MORE WE CAN OFFER!

```
p14_6     move.w    d6,d4
          beq       p14_0                to 1
          movea.l   sp4_e,a1
          lea       (a3,d3.w),a0
          lea       (a0,d5.w),a0         A0 -> end of B
          bsr       copy                 B -> sp1
          moveq     #1,d7
          move.w    d7,d6
p14_1     add.w     d7,d7                2, 4, 8 etc
          cmp.w     d4,d7                too far? . .
          bgt       p14_2                . . yes
          move.w    d7,d6                keep the power so far
          movea.l   sp4,a0
          movea.l   a0,a1
          movea.l   sp3,a2               for answer
          bsr       muls                 square again
          bne       p14_9      ---->     OOPS!
          movea.l   sp3_e,a0
          movea.l   sp4_e,a1
          bsr       copy                 copy sp3 to sp4
          bra       p14_1                next power of 2
p14_2     sub.w     d6,d4                remaining multiplications
          bra       p14_3

p14_4     lea       (a3,d3.w),a0         A0 -> B
          movea.l   sp4,a1               answer so far
          movea.l   sp3,a2               mult -> sp3
          bsr       muls
          bne       p14_9      ---->     OOPS!
          movea.l   sp3_e,a0
          movea.l   sp4_e,a1
          bsr       copy                 update answer
p14_3     dbf       d4,p14_4             count extra mults

; Put answer to A
p14_5     lea       (a3,d2.w),a1
          lea       (a1,d5.w),a1         A1 -> A end
          movea.l   sp4_e,a0
          bsr       copy                 Set answer
          bsr       p14_10               Reset B if needed
          btst      #17,d6               negate answer? . .
          beq       pend                 . . no
          lea       (a3,d2.w),a0         restore B
          bsr       neg
p14_8     bra       pend

; Power is 1
p14_0     movea.l   sp4,a0
          bsr       clrnm
          move.w    #1,-2(a0)            1 in sp4
          bra       p14_5

; Error exit
p14_9     bsr       p14_10               reset B if needed
          bra       bad_exit1 ---->

; This restores B to negative if needed and sets D0
p14_10    moveq     #0,d0
          btst      #16,d6
          bne       p14_11
          rts
p14_11    lea       (a3,d3.w),a0
          bra       neg

**************
* SIZE (p15) *
**************
; p15 sets sze in ans1

p15       lea       ans1,a0
          move.w    sze,(a0)
```

```
          moveq     #0,d0
          rts

***************
* RANGE (p16) *
***************
; p16 sets totn to ans1

p16       lea       ans1,a0
          move.w    totn,(a0)
          moveq     #0,d0
          rts

***************
* COUNT (p17) *
***************
; p17 subtracts 1 from A and returns 0 in ans1 if zero

p17       bsr       set_ad
          movea.l   sp4,a0
          bsr       clrnm
          move.w    #1,-2(a0)           set sp4 = 1
p17_1     lea       (a3,d2.w),a1
          lea       (a1,d5.w),a1        to end of A
          bsr       subnm               subtract 1 from A
          movea.l   a1,a0               set A to A0
          bra       p9_2

****************
* COUNTB (p18) *
****************
; p18 subtracts B from A and sets the result of tstnm in ans1

p18       bsr       set_ad
          lea       (a3,d3.w),a0
          lea       (a0,d5.w),a0        -> end of B
          bra       p17_1

; At end set ans1 to error code
bad_exit1 moveq     #-15,d0

; Exit setting error code from D0 in ans1
pend      lea       ans1,a0
          move.w    d0,(a0)
          moveq     #0,d0
          rts

******************
* END of PROGRAMS *
******************


***************
* Subroutines *
***************
; set_ad converts D2 - D4 to relative addresses of A, B and C
; It also sets D5.W to the number length in bytes
; and sets A3 to adr1

set_ad    move.w    sze,d5
          lsl.w     #2,d5               number of bytes
          mulu.w    d5,d2               convert to . .
          mulu.w    d5,d3               . . relative . .
          mulu.w    d5,d4               . . addresses
          movea.l   adr1,a3
          rts

-----------------------------------------------------------

; getsp returns in A0 the address of an area of D1.L bytes
```

```
gsp_reg   reg       d2-3/a1-3
getsp     movem.l   gsp_reg,-(sp)
          moveq     #-1,d2              this job
          moveq     #mt_alchp,d0
          trap      #1
          tst.l     d0
          movem.l   (sp)+,gsp_reg
          rts
```

```
; relsp releases space and clears both sze and adr0
; no registers save D0 are used.
; On exit D0 = 0 and condition codes are set

relsp_reg reg       d1-3/a0-3
relsp     movem.l   relsp_reg,-(sp)
          lea       adr0,a1
          movea.l   (a1),a0
          clr.l     (a1)
          lea       sze,a1
          clr.w     (a1)
          move.l    a0,d0
          beq       relsp1
          moveq     #mt_rechp,d0
          trap      #1
relsp1    movem.l   (sp)+,relsp_reg
          moveq     #0,d0
          rts
```

```
; set_sgn sets absolute values of B and C and codes D5.TOP to signal
; signs for DIV, MUL and MOD

; Code B C
;  00  + +
;  01  - -
;  10  + -
;  11  - +

set_sgn   swap      d5
          clr.w     d5
          tst.l     (a3,d3.w)           B
          bpl       set_sgn1            +
          addq.w    #8,d5               mark -B
          lea       (a3,d3.w),a0
          bsr       neg                 negate B
          cmp.w     d3,d4               B = C? . .
          beq       set_sgn6            . . yes
set_sgn1  tst.l     (a3,d4.w)           C
          bpl       set_sgn2            +
          lea       (a3,d4.w),a0
          bsr       neg                 negate C
set_sgn6  addq.w    #4,d5               mark -C
set_sgn2  subq.w    #4,d5               just -C? . .
          bne       set_sgn3            . . . no
          addq.w    #2,d5               mark B/-C
set_sgn3  subq.w    #4,d5               just -B? . .
          bne       set_sgn4            . . . no
          addq.w    #3,d5               mark -B/C
set_sgn4  subq.w    #4,d5               -B and -C . .
          bne       set_sgn5            . . no
          addq.w    #1,d5
set_sgn5  swap      d5
          rts
```

```
; rst_sgn (reset sign) negates B and C if need be.
```

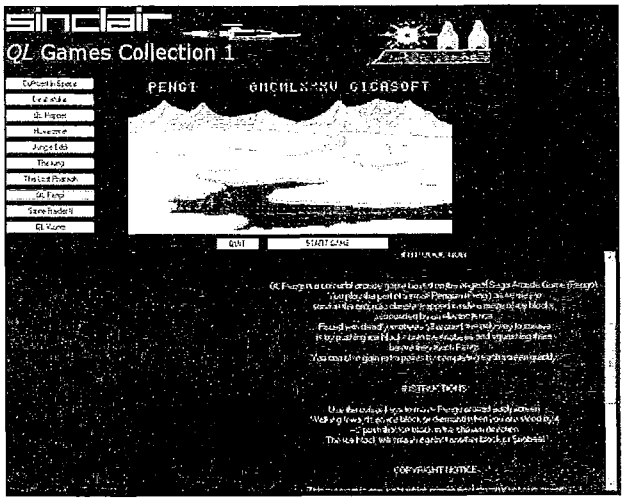
```
rst_sgn   move.l    d5,d7
          swap      d7
          andi.w    #3,d7              keep the two-bit code
          bne       rst_sgn1           something to do
rst_sgn3  rts
rst_sgn1  cmp.w     d3,d4              B = C? . .
          bne       rst_sgn4           . . no
          cmpi.w    #1,d7
          bne       rst_sgn5
          addq.w    #2,d7              set 01 to 11
          bra       rst_sgn4

rst_sgn5  cmpi.w    #2,d7
          bne       rst_sgn4
          clr.w     d7
rst_sgn4  subq.w    #1,d7              1? . .
          bne       rst_sgn2           . . no
          bsr       b_neg              negate B
          bra       c_neg              negate C and return

rst_sgn2  subq.w    #1,d7
          bne       b_neg              3 so negate B

c_neg     cmp.w     d3,d2
          beq       rst_sgn3
          lea       (a3,d4.w),a0       negate C if not equal to A
          bra       neg

b_neg     cmp.w     d4,d2
          beq       rst_sgn3
          lea       (a3,d3.w),a0       negate B if not equal to A
          bra       neg
```

```
; LB4 decimalises a number
; D3 is negative for a negative number.
; A0 -> the absolute value of the number
; The digits are found from the least significant end and
; placed backwards in the name space

LB4       movea.l   a0,a5
          MOVEA.L   nme_e,A2  -> end of name space
          MOVEQ     #10,D2
LB0       MOVE.w    sze,D0
          add.w     d0,d0     number of words in number . .
          subq.w    #1,d0     . . -1
          MOVEQ     #0,D1
LB1       MOVE.W    (A0)+,D1  Next word
          DBNE      D0,LB1    Find 1st non-zero
          BEQ.S     LB3       Finished
LB2       DIVU      D2,D1     Divide by 10
          MOVE.W    D1,-2(A0) Set remainder at start of next word
          MOVE.W    (A0)+,D1  Get next word
          DBF       D0,LB2    Count sze*2 words

; Here D1.TOP contains the next decimal digit

          SWAP      D1
          ADDI.B    #'0',D1
          MOVE.B    D1,-(A2)  Store digit
          MOVEA.L   sp4,A0    Reset pointer to number
          BRA       LB0

LB3       TST.B     D3        Negative? . .
          BEQ       LB5       . . no
          MOVE.B    #'-',-(A2)
LB5       rts
```

QL Games Collection 1 - Review

by Peter Scott

QL gaming fans are spoilt for choice at the moment with the steady supply of re-released games from RWAP software. The latest tasty morsel is something a bit different.



Sinclair QL Games Collection 1 is a compilation of 10 classic arcade style games. The collection consists of a single installation file with a built in runtime engine based on Q-emuLator by Daniele Terdina. The games are selected with a single button and there is no need for any extra settings or configuration.

Once installed the collection runs showing a tidy menu screen which has pictures of each game plus the instructions, each game runs with a single click and when closed returns back to the menu. The collection consists of 10 games with a wide choice of game play. The games are:

Cuthbert in Space - a single screen shoot 'em up which has you fending off aliens while collecting fuel and treasure.

Deathstrike - a slick scrolling shooter based on Scramble with enemy installations to shoot while collecting fuel.

QL Hopper - a Frogger clone with a frog needing to be helped across a busy road to safety.

Hoverzone - a fast paced scrolling shooter where aliens try to capture people from the surface.

Jungle Eddi - a colourful pretty multi screen adventure which sees you leaping about to escape the jungle.

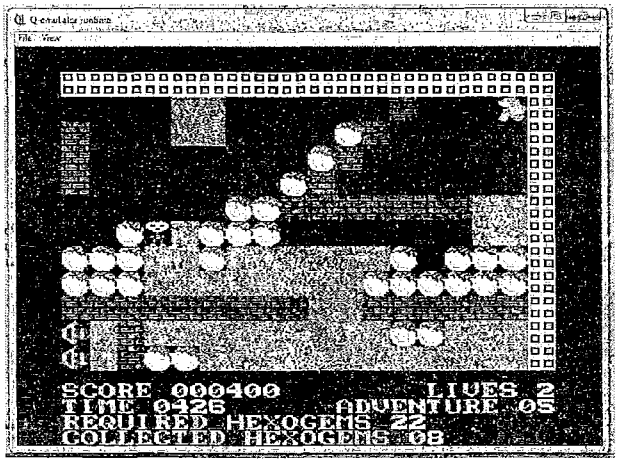
The King - a Donkey Kong based game (no not Elvis) in which you leap over thrown barrels to save the girl from a giant ape.

The Lost Pharaoh - a speedy maze game with a lost tomb to be explored and treasure to be found.

QL Pengi - Pengo style maze game where a cute penguin has to survive each level by crushing the baddies with sliding blocks of ice.

Stone Raider II - Boulder Dash based head scratcher which sees you digging paths to collect diamonds while avoiding traps and rocks.

QL Vroom - F1 style racing game similar to Pole Position.



Each game runs at a good pace and modern high resolution displays are provided for with the addition of a graphics filter to avoid pixilation at high resolutions. Most of the games were new to myself and I was surprised by how crisp they looked on my PC with bold bright colours.

Being a PC only program there is no way to play the games on an original QL but the compilation has been developed to showcase the QL and its games to people who will be unfamiliar with the system. That said at £10 (£1 per game) the compilation is superb value for money and should appeal to existing QL fans as well as potential new users.

Using the Parallel Printer Port

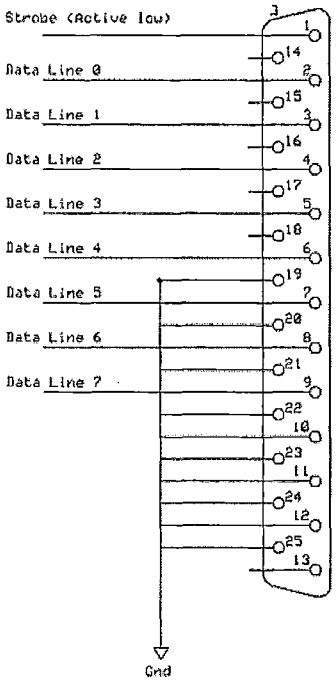
by Ian Burkinshaw

In part 4 of my series, I2C Interface for QL Emulators, I have shown how you can use an LCD Display using the PCF8574 device with an RS232 serial to I2C converter that provides an 8 bit parallel I/O port. However you can have a simple 8 bit only output port from QXL, QPC2, Qemulator and Super Gold Card equipped QLs. So a LCD display can be used in a similar manner. Or for that matter any other application that requires a simple output port. This is what I will cover in this article and also expand a little more on using LCD displays, which can also be applied to I2C users. I should point out I have not been able to get this to work with Qlay or QL2K. I have not spent a great deal of time looking at these two emulators in this regard, so it is not to say, that with some effort they cannot be made to work. But the systems I have tested and know work will, I think, cover most QL/SMSQ system users.

Let me start with QXL, QPC2 and Qemulator users, since this is the simplest, because in all these cases we are using a PC's hardware, which works in a slightly different manner to the Super Gold Card's printer port as far as implementing a parallel output port is concerned.

First your PC must have a parallel printer port. This will be a 25 way female D-Type connector on your machine. If you do not have one, then a USB to parallel converter may work. However if you use one of these, do ensure that in the Device Manager of your PC operating system, under the Ports (Com & LPT) shows 'LPT1'. Not all do.

So for QXL, QPC2 and Qemulator users you only need the following interface.

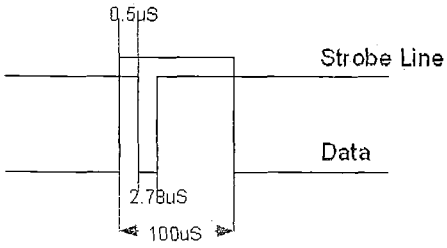


I have shown the strobe line, but in the case of PC based systems this is not required. The grounding of the various control lines gives us latched outputs. The following listing will make the data lines go up and down. If you put an LED, anode to the data line and via a resistor of say 470ohm to ground, the LED will flash. To enable the 'PAR' device to work you must ensure TK2 is running.

```
10 TK2_EXT
20 OPEN#3;par
30 REPeat loop
40 PRINT#3;CHR$(255);
50 PAUSE 25
60 PRINT#3;CHR$(0);
70 PAUSE 25
80 END REPeat loop
```

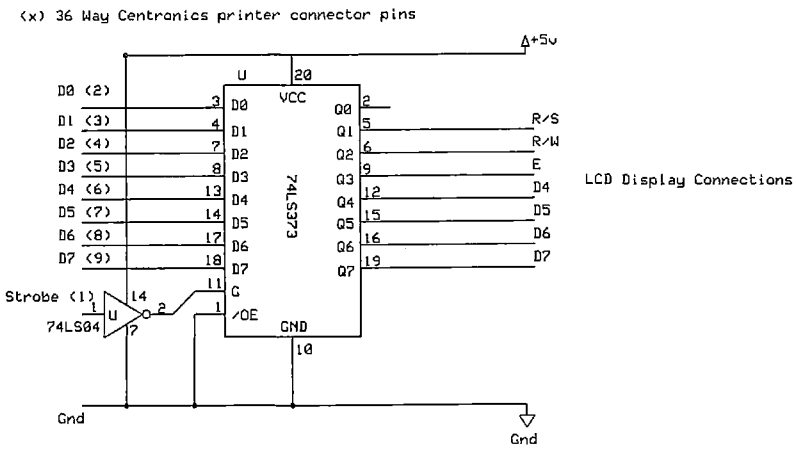
If you are a Super Gold Card user then things get a little more complicated. The parallel printer port on the Super Gold Card only has the data lines and the strobe line. It does not have the other control lines that a standard PC parallel printer port has. So it will not provide the latched output that the PC parallel port can. However having the strobe line we only need to add a latching circuit.

The signal timing from the SGC is shown in the graphic to the right:



Output from Super Gold Card

So the data lines are set, and the flip flops in the latch are then clocked by the strobe line. This latches the output from the flip flops low or high as required and hold that state until another data and strobe event take place. The circuit to achieve this is shown on the next page.



The 74LS373 contains 8 flop flops with a common clock (G), so that any data pattern will be latched on to the output when trigged by the strobe signal. The strobe signal is inverted, hence the 74LS04 inverter chip. The 74LS04 has 6 inverters, in this case only one is used.

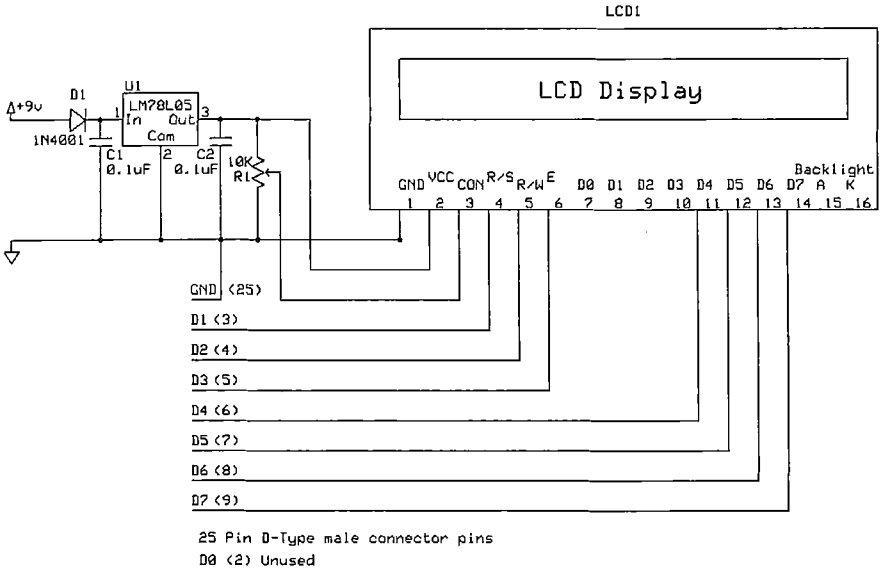
I have shown the 36 pin centronics connector pin numbers so you can use the printer cable that came with your SGC. You will need a female centronics connect to be able to use this. These are available from the usual electronic component suppliers If you look in the SGC manual it shows the IDE pin header connections. This is another way of connecting to the SCG.

No power is available from parallel ports, unlike the USB to I2C converters, so arrangements have to be made to provide 5V to this circuit. When we move on to driving LCD displays, you will see such an arrangement.

To test this circuit the same listing as shown above for the PC parallel port can be used. Again you can connect LED(s) in the same way. So anode to the output data line (Q0–Q7), then cathode to ground via a resistor (470 ohm).

So now to connecting a LCD display.

I have shown the 25 way D-Type connector pin numbers if you are using a PC parallel port, and the data line numbers (D1 to D7) for the SGC latch circuit. So if you are using a PC, combine this circuit with the 25 way D-Type circuit at the beginning of this article.



As you may see in the diagram to the right, there is a 5V regulator(U1), this is used to supply the LCD display itself but can be also used to supply 5V to the SGC latch circuit. You can take the power for this from pin 3 (out) of U1.

So that deals with the hardware, but what about the software? Find below a listing which is common to all the following routines to show what can be done with a LCD display. This is not too different from the listing in the last I2C article, but has been changed for parallel port use.

```
10 REMark Parallel LCD Experiments common routines
20 REMark EPE Feb 1997, How to use intelligent LCD's
30 init
40 init_LCD
50 clear_LCD
900 PAUSE
```

```
910 clear_LCD
920 PRINT "End"
940 PRINT#3;CHR$(0);
950 CLOSE#3
960 STOP
970 :
1000 DEFine PROCedure init
1010 rs=2:rw=4:en=8:REMark rs is register select, rw is read/write (only needed for using
     display ram)
1020 OPEN#3;par
1030 CLS
1040 PRINT#3;CHR$(0);:REMark to set all parallel data lines to 0
1050 FOR r=1 TO 3:REMark This FOR NEXT loop is to ensure the LCD is in it initial mode which is
     8 bit interface mode, after power up. This does not reset the LCD Display.
1060 load_LCD 48,0
1070 NEXT r
1080 PRINT "Ensured LCD Display in 8 bit interface mode."
1090 PRINT#3;CHR$(0);:REMark to set all parallel lines to 0
1100 END DEFine init
1110 :
1120 DEFine PROCedure init_LCD
1130 PRINT#3;CHR$(0);:REMark to set all parallel line to 0
1140 load_LCD 32,0:REMark Sets LCD to 4 bit interface mode
1150 PRINT "Set LCD to 4 bit interface mode."
1160 load_LCD 32,0:REMark LCD Function set to 2 line mode, 1st Nibble
1170 load_LCD 128,0:REMark LCD Function set to 2 line mode, 2st Nibble
1180 REMark Note, be careful not to set D4(16) when addressing the Function Set register, since
     this will return the LCD to 8 bit interface mode.
1190 PRINT "Function Set to 2 line mode."
1200 load_LCD 0,0:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, this is the Display
     On/Off & Cursor register, 1st Nibble
1210 load_LCD 240,0:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, this is the
     Display On/Off & Cursor register, 2nd Nibble
1220 PRINT "Set LCD to Display On, Cursor On and Cursor Blinking."
1230 load_LCD 0,0:REMark Sets LCD to Character Entry mode, Increment Cursor Position No Display
     shift, 1st nibble.
1240 load_LCD 96,0:REMark Sets LCD to Character Entry mode, Increment Cursor Position No Display
     shift, 2st nibble.
1250 PRINT "Set LCD to Increment Cursor Position and No Display shift."
1260 END DEFine init_LCD
1270 :
1280 DEFine PROCedure LCD_message(message$)
1290 mlen=LEN(message$)
1300 FOR mc=1 TO mlen
1310 ms$=message$(mc)
1320 ms=CODE(ms$)
1330 nib1=(INT(ms/16))
1340 nib2=ms-(nib1*16)
1350 nib1=(nib1*16):REMark 1st nibble
1360 nib2=(nib2*16):REMark 2nd nibble
1370 load_LCD nib1,rs
1380 load_LCD nib2,rs
1390 PRINT ms$;" ASCI Character Number:";ms;" First Nibble:";nib1;" Second Nibble:";nib2
1400 NEXT mc
1410 END DEFine LCD_message
1420 :
1430 DEFine PROCedure move_second_line
1440 load_LCD 192,0:REMark Move to start of second line 1st nibble.
1450 load_LCD 0,0:REMark Move to start of second line 2st nibble.
1460 PRINT "Moved to second line on LCD"
1470 END DEFine move_second_line
1480 :
1490 DEFine PROCedure load_LCD(lcd_data,rsm)
1500 PRINT#3;CHR$(rsm);
1510 PRINT#3;CHR$(en+rsm);
1520 PRINT#3;CHR$(lcd_data+en+rsm);
1530 PRINT#3;CHR$(lcd_data+rsm);
1540 REMark PAUSE:REMark Use this to step though the workings on the LCD.
1550 END DEFine load_LCD
1560 :
```

```
1570 DEFine PROCedure more_characters
1580 PRINT "Press any key to see more of the character set              "
1590 PAUSE
1600 clear_LCD
1610 END DEFine more_characters
1620 :
1630 DEFine PROCedure clear_LCD
1640 load_LCD 0,0:REMark clear LCD and return cursor to start, 1st nibble.
1650 load_LCD 16,0:REMark clear LCD and return cursor to start, 2st nibble.
1660 load_LCD 128,0:REMark reset display address to 0, 1st nibble.
1670 load_LCD 0,0:REMark reset display address to 0, 1st nibble.
1680 PAUSE 10:REMark giving time for display to respond to the last commands
1690 END DEFine clear_LCD
32000 DEFine PROCedure UPDATE
32010 SAVE win1_parallel_LCD_common_bas
32020 PRINT "Update Complete"
32030 END DEFine UPDATE
```

The common routine is annotated, so you can follow what is going on. Now the first experiment which will have the display, "QLToday Forever" across two lines of the display. Assuming you are using a display with at least 2 lines.

```
60 :
70 LCD_message "QLToday"
80 move_second_line
90 LCD_massage "Forever"
100 :
```

Now in the second experiment, here we will make the LCD display all the character set. This is not the most compact program. But I have written this deliberately so you see what is going on. That some characters use standard ASCII code, but others do not. So shows you how to display the Chinese and Greek characters. Now I guess not many will need Chinese, but the symbols and Greek are useful.

```
60 LCD_message " !#$%&'()*+,-/:;"
70 move_second_line
80 LCD_message "<=>?0123456789"
90 more_characters
100 LCD_message "ABCDEFGHIJKLMNOP"
110 move_second_line
120 LCD_message "QRSTUVWXYZ"
130 more_characters
140 LCD_message "abcdefghijklmnop"
150 move_second_line
160 LCD_message "qrstuvwxyz"
170 more_characters
180 PRINT "Now the other characters"
190 FOR c=91 TO 96:LCD_message CHR$(c):NEXT c
200 move_second_line
210 FOR c=123 TO 127:LCD_message CHR$(c):NEXT c
220 more_characters
230 PRINT "Chinese Characters":REMark Now the LCD will display Chinese characters
240 FOR c=160 TO 175:LCD_message CHR$(c):NEXT c
250 move_second_line
260 FOR c=176 TO 191:LCD_message CHR$(c):NEXT c
270 more_characters
280 FOR c=192 TO 207:LCD_message CHR$(c):NEXT c
290 move_second_line
300 FOR c=208 TO 223:LCD_message CHR$(c):NEXT c
310 more_characters
320 PRINT "Greek Characters":REMark Now the LCD will display Greek Characters"
330 FOR c=224 TO 239:LCD_message CHR$(c):NEXT c
340 move_second_line
350 FOR c=240 TO 255:LCD_message CHR$(c):NEXT c
360 PRINT "All characters have now been displayed, press any key to clear LCD display and end
program."
```

This third experiment demonstrates display scrolling.

```
60 a$="This demonstrates the LCD display scroll":REMark this line is 40 characters long, the
      capacity of one line on the LCD display.
70 LCD_message a$
80 mlen=LEN(a$)
```

```
90 PAUSE 100:REMark sets the delay before the display scrolls.
100 FOR c=1 TO (mlen-16):REMark assuming a 16 character per line display.
110 load_LCD 16,0:REMark Set scrolling one character position to the left, 1st nibble
120 load_LCD 128,0:REMark Set scrolling one character position to the left, 2nd nibble
130 PAUSE 25:REMark Sets the speed the display scrolls.
140 NEXT c
```

The fourth and last experiment demonstrates setting up user defined graphics. The first 7 locations of the character set are user definable. This routine just addresses this first character with a stick man. Again I have written this deliberately so you see what is going on.

```
60 load_LCD 128,0:REMark set display address to 0, 1st nibble
70 load_LCD 0,0:REMark set display address to 0, 2nd nibble.
80 FOR a=0 TO 7
90 LCD_message CHR$(a)
100 NEXT a
105 PAUSE
110 load_LCD 64,0:REMark set CGRAM address to 0, 1st nibble
120 load_LCD 0,0:REMark set CGRAM address to 0, 2nd nibble.
130 load_LCD 0,rs:REMark set character 1 address 0, 1st nibble
140 load_LCD 224,rs:REMark set character 1 address 0, 2nd nibble.
150 load_LCD 16,rs:REMark set character 1 address 1, 1st nibble
160 load_LCD 16,rs:REMark set character 1 address 1, 2nd nibble.
170 load_LCD 0,rs:REMark set character 1 address 2, 1st nibble
180 load_LCD 224,rs:REMark set character 1 address 2, 2nd nibble
190 load_LCD 0,rs:REMark set character 1 address 3, 1st nibble
200 load_LCD 64,rs:REMark set character 1 address 3, 2nd nibble
210 load_LCD 16,rs:REMark set character 1 address 4, 1st nibble
220 load_LCD 16,rs:REMark set character 1 address 4, 2nd nibble
230 load_LCD 0,rs:REMark set character 1 address 5, 1st nibble
240 load_LCD 64,rs:REMark set character 1 address 5, 2nd nibble
250 load_LCD 0,rs:REMark set character 1 address 6, 1st nibble
260 load_LCD 160,rs:REMark set character 1 address 6, 2nd nibble
270 load_LCD 16,rs:REMark set character 1 address 7, 1st nibble
280 load_LCD 16,rs:REMark set character 1 address 7, 2nd nibble
285 PAUSE
290 clear_LCD
300 FOR a=0 TO 7
310 LCD_message CHR$(a)
320 NEXT a
```

You should see the display displaying rubbish to start with, this is normal. Once the routine has loaded the user defined graphic, the routine will display the first 7 character locations again, but the first one will now show the stick man.

As I said before, these routines will work with the I2C LCD common routine published in my last article.

Until next time have fun.

References

Farnell - **http://uk.farnell.com/jsp/home/homepage.jsp**

RS - **http://uk.rs-online.com/web/**

PCF8574(A) Data Sheet
http://www.nxp.com/documents/data_sheet/PCF8574.pdf
http://focus.ti.com/lit/ds/symlink/pcf8574.pdf

How to use intelligent LCD's Part 1, Everyday Practical Electronics, February 1997
Downloadable from http://www.wizard.org/auction_support/lcd1.pdf

How to use intelligent LCD's Part 2, Everyday Practical Electronics, March 1997
Downloadable from http://www.wizard.org/auction_support/lcd2.pdf

Character LCD Displays – Part 1
http://www.protostack.com/blog/2010/03/character-lcd-displays-part-1/

Hitachi LDC Driver chip, advanced stuff in here. But does show all the capabilities of LCD displays that use this chip - **http://www.sparkfun.com/datasheets/LCD/HD44780.pdf**

[Ed. The Chinese characters that Ian Burkinshaw refers to are in fact Japanese.]

Programming in Assembler, Part 31
LibGen - Library Generator - Part 1

by Norman Dunbar

Introduction

In the last issue, I looked at the creation and use of libraries with gwasl.

Elsewhere in this issue you will hopefully find a few comments from George on the matter and a bit of an email conversation we had about how to work around the problem of only exposing the required routines in the libraries we create without exposing the internal working routines.

For example, in my small example library, I would only expose the various CLEAR_xxx routines, but not the internal JUST_DO_IT routine. Also, it is not required to expose any of the equates used internally by the library – simply because they may conflict with your own equates used elsewhere in the program and because, by the time the library is assembled, the equates have been converted into absolute values anyway. George mentioned editing the library source code, setting a new equate with a given prefix, to the routines you wish to expose, similar to the following:

```
LB_CLEAR_SCREEN equ clear_screen
```

In the above, you can see that I've set LB_CLEAR_SCREEN equal to CLEAR_SCREEN.

When the symbol file is converted to text, it will have the following in it:

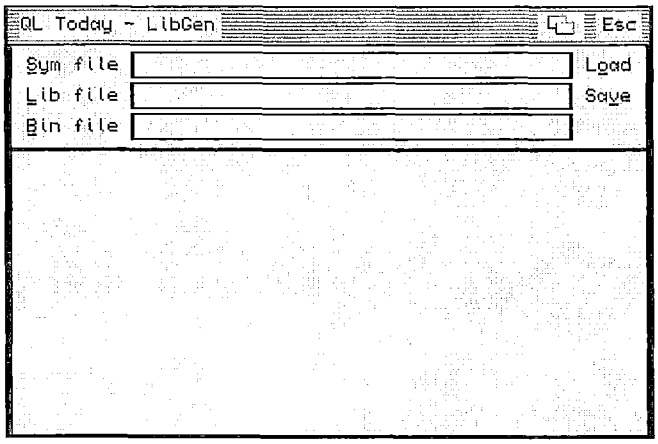
```
...
CLEAR_SCREEN     EQU     *+$00000000
LB_CLEAR_SCREEN  EQU     *+$00000000
JUST_DO_IT       EQU     *+$00000012
...
```

You can see that CLEAR_SCREEN and LB_CLEAR_SCREEN are the same. George proposed that a SuperBasic program could be quickly written to extract only those equates prefixed by, for example, 'LB_' and this extracted file could then be used to expose only the chosen routines. I liked this idea, but I'm sort of wary about having to edit my source code and add extra equates whenever I write a new routine.

Equally, this is an assembler tutorial and at present I'm writing about the Pointer Environment, so how about a useful – yes, the first one – utility to read in a sym_lst file, display all the code offset equates (and only those) and when the user has selected the desired items, write out a valid file that can be included. This issue's article is that very utility!

LibGen

I've chosen LibGen as my utility name. The finished screen will hopefully look very similar to the following screen shot:



Along the top is a green/white (paper 92) caption bar which is simply an information window.

The actual title is itself embedded in another small information window with white paper.

The move loose item and the Esc loose items are embedded within the main caption bar. To keep code sizes to a minimum, there is no Size or Sleep loose items in this utility.

Moving down the screen there is a large information window with a black border and white paper containing the 5 loose items – Sym file, Lib file, Bin file, Load and Save, and three further information Windows, each with white paper and a black border.

The largest part of the window is taken up by an application window at the bottom. This too has white paper and a black border. This is where we will hold the code offset lines from the symbol file.

In operation, you click on "sym file" to allow the name of the sym_lst file in question to be edited. If the edit succeeds, the "_sym_lst" extension is removed and replaced with "_lib" for the "Lib file" and "_bin" for the "Bin file".

You can, however, edit these auto-generated names by hitting the appropriate loose item. When happy with the names, hit the "Load" loose item to read in the file. The application window will fill up with appropriate options from the file and you will be able to choose the ones you wish to keep.

By default everything is selected, you only need to deselect the ones you don't want to keep. I'm working on the assumption that you will want to keep more than you don't, so it should be easier to deselect the few than select the many.

Once happy with your selection, hit the "Save" loose item and the selected entries will be copied to the "Lib file" along with a line to include the "Bin file", as the following example illustrates:

```
CLEAR_SCREEN  EQU     *+$00000000
CLEAR_TOP     EQU     *+$00000004
CLEAR_BOTTOM  EQU     *+$00000008
CLEAR_TO_EOL  EQU     *+$0000000C
CLEAR_LINE    EQU     *+$00000010

              lib     win1_gwasl_libs_lib_cls_bin
```

Now all you need to do is add one line to your program's source:

```
              in      win1_gwasl_libs_lib_cls_lib
```

Hopefully, this is a lot easier than editing source code, adding extra equates, and running a SuperBasic program to extract them and so on.

Window Design

Make sure that you download the latest versions of SETW and Easy PEasy from **http://gwiltprogs.info/page2.htm** as George has recently updated these to make our lives much easier. We will be using the new features of Easy PEasy in this utility. We begin by designing our window with SETW. Now, contrary to my own instructions above, I have to declare here that I'm not yet using the latest version of SETW! I'm also abbreviating the prompts etc in the following steps to try and keep coding to a minimum.

1. Enter a name, I used libgenwin for mine, but feel free to make up your own.
2. Enter the following text objects by pressing the 'N' key, then typing in the text:
   - QL Today – LibGen
   - Esc
   - Sym file
   - Lib file
   - Bin file
   - Load
   - Save

   Press ESC when done.
3. For the sprites, blobs and patters, simple press Esc as we have none of those.
4. There is one single main window.
5. There are 7 loose items within it.

6. There are 6 information windows.

7. Information windows 1 to 4 have no objects, number 5 has one, and number 6 has none.

8. There will be one application window, but it has no menu items. (We will build the menu dynamically.)

9. The main window has the following attributes:

- Shadow size 2
- Border size 1, border colour is QL Black.
- Paper colour is QL White.
- Select the default arrow sprite for the pointer.

10. When prompted, twice, to select default settings for loose items, chose 'N' both times. We will enter our own settings.

11. The current item border size is 1 and the colour is QL Black.

12. Unavailable attributes are:

- Background colour is QL White.
- Ink colour is QL Grey.

13. Available attributes are:

- Background colour is QL White.
- Ink colour is QL Black.

14. Selected attributes are:

- Background colour is QL Green.
- Ink colour is QL Black.

15. For the 7 loose items, set them up as follows:

1. Type is text, choose the 'Esc' text, no selection key.
2. Press the down arrow key to change "Text" to "Sprite". In the next screen, press the down arrow to highlight "Move (sprite)" and

   Press Enter.
3. Type is text, choose the 'Sym file' text, selection key is 'S' and underlined.
4. Type is text, choose the 'Lib file' text, selection key is 'L' and underlined.
5. Type is text, choose the 'Load' text, selection key is 'O' and underlined.
6. Type is text, choose the 'Save' text, selection key is 'V' and underlined.
7. Type is text, choose the 'Bin file' text, selection key is 'B' and underlined.

16. For the 6 information windows, set them up as follows:

1. Border size zero, paper QL 92.
2. Border size 1, colour QL Black, paper QL White.
3. Border size 1, colour QL Black, paper QL White.
4. Border size 1, colour QL Black, paper QL White.
5. Border size 0, paper QL White. When prompted for an object, choose Text "QL Toady – LibGen", set the ink to QL Black and the two CSizes to zero.
6. Border size 1, colour QL Black, paper QL White.

17. The application window needs the following:

- Border size 1.
- Colour QL Black.
- Paper QL White.
- Choose the standard arrow sprite.
- Set the selection key to TAB.

18. There now follows a session of pressing arrow keys and F2 etc to size and position all the loose items, information windows to build our window. The main window itself is first:

- The size is 336 by 224.
- The window is not a variable one.
- The pointer origin is 50 by 37.

19. The 7 loose items should be sized and positioned as follows:

1. 24 by 13 at 308 by 3.
2. 24 by 13 at 278 by 3.
3. 50 by 14 at 8 by 23.
4. 50 by 14 at 8 by 39.
5. 28 by 14 at 300 by 23.
6. 28 by 14 at 300 by 39.
7. 50 by 14 at 8 by 55.

20. The 6 information windows should be sized and positioned as follows:

1. 336 by 20 at 0 by 0.
2. 332 by 53 at 2 by 20.
3. 228 by 12 at 66 by 24.
4. 228 by 12 at 66 by 40.
5. 106 by 13 at 6 by 3. When prompted for the object, position it at 0 by 1.
6. 228 by 12 at 66 by 56.

21. The application window should be 332 by 148 and positioned at 2 by 75.

You will notice that the screen looks a bit cluttered and the program prompts can be hard to see under all the loose items, information windows etc.

When you are finished, the window will be displayed on screen. If it looks OK, all well and good, if not, don't worry, it can be fixed in the generated code, which should look as follows. We need to do some editing as well, but I'll cover that below.

```
SYS_SPR dc.w    0,1,2,3,4,5,6,7,8,9,10,11,12,13
        dc.w    14,15,16,17,18,19,20,21,22,23,24
        dc.w    25,26,27,28,29,30,31,32,33,34,35
        dc.w    36,37

txt0    dc.w    txt0_e-2-txt0
        dc.b    "QL Today - LibGen"
txt0_e  ds.b    0
        ds.w    0

txt1    dc.w    txt1_e-2-txt1
        dc.b    "Esc"
txt1_e  ds.b    0
        ds.w    0

txt2    dc.w    txt2_e-2-txt2
        dc.b    "Sym file"
txt2_e  ds.b    0
        ds.w    0

txt3    dc.w    txt3_e-2-txt3
        dc.b    "Lib file"
txt3_e  ds.b    0
        ds.w    0

txt4    dc.w    txt4_e-2-txt4
        dc.b    "Bin file"
txt4_e  ds.b    0
        ds.w    0

txt5    dc.w    txt5_e-2-txt5
        dc.b    "Load"
txt5_e  ds.b    0
        ds.w    0

txt6    dc.w    txt6_e-2-txt6
        dc.b    "Save"
txt6_e  ds.b    0
        ds.w    0

app_list0
        dc.w    appw0-*
        dc.w    0

appw0
        dc.w    332         xsize
        dc.w    148         ysize
        dc.w    2           xorg
        dc.w    75          yorg
        dc.w    0           flag
        dc.w    1           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    0           pspr *
        dc.w    0           setr *
        dc.w    0           draw *
        dc.w    ahit0-*     hit *
        dc.w    0           cntrl *
        dc.w    0           nxsc
```

```
        dc.w    0           nysc
        dc.b    9           skey
        dc.b    0           spr1

pobl4
        dc.w    102         xsize
        dc.w    10          ysize
        dc.w    0           xorg
        dc.w    1           yorg
        dc.b    0           type
        dc.b    0           spar
        dc.l    0           spce
        dc.w    txt0-*      pobj *
        dc.w    -1

infw0
        dc.w    336         xsize
        dc.w    20          ysize
        dc.w    0           xorg
        dc.w    0           yorg
        dc.w    0           flag
        dc.w    0           borw
        dc.w    0           borc
        dc.w    92          papr
        dc.w    0           pobl *

        dc.w    332         xsize
        dc.w    53          ysize
        dc.w    2           xorg
        dc.w    20          yorg
        dc.w    0           flag
        dc.w    1           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    0           pobl *

        dc.w    228         xsize
        dc.w    12          ysize
        dc.w    66          xorg
        dc.w    24          yorg
        dc.w    0           flag
        dc.w    1           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    0           pobl *

        dc.w    228         xsize
        dc.w    12          ysize
        dc.w    66          xorg
        dc.w    40          yorg
        dc.w    0           flag
        dc.w    1           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    0           pobl *

        dc.w    106         xsize
        dc.w    13          ysize
        dc.w    6           xorg
        dc.w    3           yorg
        dc.w    0           flag
        dc.w    0           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    pobl4-*     pobl *

        dc.w    228         xsize
        dc.w    12          ysize
        dc.w    66          xorg
        dc.w    56          yorg
        dc.w    0           flag
        dc.w    1           borw
        dc.w    0           borc
        dc.w    7           papr
        dc.w    0           pobl *

        dc.w    -1          end
```

So far, so good. Nothing in the above should differ from what you typed into SETW. If anything has been created in error, you can adjust it in the above.

Coming next are the loose items definitions and it is here that we have to change a few things.

In the first loose item below, the Selection Key has been set to 3. When we created this loose item in SETW, we didn't select a selection key for it. The first loose item is for Esc, and we give it a selection key of 3, which corresponds to the Cancel event code.

The second of the loose items needs the selection key to be set to the event code of 5 for Move. (All changes below have comments.)

Note: When using a system sprite, the sprite pointer has to point, relatively, at a word containing the sprite number.

```
litm0
        dc.w    24,13       xsize, ysize
        dc.w    308,3       xorg, yorg
        dc.b    0,0         xjst, yjst
        dc.b    0,3         type, skey        SKEY = ESC event.
        dc.w    txt1-*      pobj *
        dc.w    0           item
        dc.w    afun0_0-*   pact *

        dc.w    24,13       xsize, ysize
        dc.w    278,3       xorg, yorg
        dc.b    0,0         xjst, yjst
        dc.b    2,5         type, skey        SKEY = MOVE event. TYPE = sprite.
        dc.w    sys_spr+12-*  pobj *          POBJ = move sprite.
        dc.w    1           item
        dc.w    afun0_1-*   pact *
```

The next two loose items have their horizontal justification set to -1 which means "justify right". Again, the changes from the code generated by SETW is highlighted in the comments.

```
        dc.w    50,14       xsize, ysize
        dc.w    8,23        xorg, yorg
        dc.b    -1,0        xjst, yjst        XJST = right justify.
        dc.b    -1,83       type, skey
        dc.w    txt2-*      pobj *
        dc.w    2           item
        dc.w    afun0_2-*   pact *

        dc.w    50,14       xsize, ysize
        dc.w    8,39        xorg, yorg
        dc.b    -1,0        xjst, yjst        XJST = right justify.
        dc.b    -1,76       type, skey
        dc.w    txt3-*      pobj *
        dc.w    3           item
        dc.w    afun0_3-*   pact *
```

The remainder of the loose items are as generated, except for loose item number 6 which also needs to be right justified. Again, the comments note where the change has been made. Everything after the loose items is as generated. Note however that I have split the "flag" word below into "flag" and "shad". SETW combines the two and generates a word for the flag byte and the shadow depth byte. I prefer to see them as they are, a pair of separate bytes. You can leave the word set to $8002 if you wish. The highest bit of the flag byte is set to clear the window.

```
        dc.w    28,14       xsize, ysize
        dc.w    300,23      xorg, yorg
        dc.b    0,0         xjst, yjst
        dc.b    -2,79       type, skey
        dc.w    txt5-*      pobj *
        dc.w    4           item
        dc.w    afun0_4-*   pact *

        dc.w    28,14       xsize, ysize
        dc.w    300,39      xorg, yorg
        dc.b    0,0         xjst, yjst
        dc.b    -3,86       type, skey
        dc.w    txt6-*      pobj *
        dc.w    5           item
        dc.w    afun0_5-*   pact *

        dc.w    50,14       xsize, ysize
        dc.w    8,55        xorg, yorg
        dc.b    -1,0        xjst, yjst        XJST = right justify.
        dc.b    -1,66       type, skey
        dc.w    txt4-*      pobj *
        dc.w    6           item
        dc.w    afun0_6-*   pact *

        dc.w    -1          end

litm1
        dc.w    16404,12    xsize, ysize
        dc.w    0,0         xorg, yorg
        dc.b    0,0         xjst, yjst
        dc.b    0,0         type, skey
        dc.w    0           pobj *
        dc.w    0           item
        dc.w    0           pact *
        dc.w    -1          end

wd0
        dc.w    336         xsize
        dc.w    224         ysize
        dc.w    50          xorg
        dc.w    37          yorg

        dc.b    0           flag              FLAG = clear the window. WAS A WORD!
        dc.b    2           shad              SHAD = shadow depth byte.
```

```
        dc.w    1       borw
        dc.w    0       borc
        dc.w    7       papr
        dc.w    0       sprt *
        dc.w    1       curw
        dc.w    0       curc
        dc.w    7       uback
        dc.w    255     uink
        dc.w    0       ublob *
        dc.w    0       upatt *
        dc.w    7       aback
        dc.w    0       aink
        dc.w    0       ablob *
        dc.w    0       apatt *
        dc.w    4       sback
        dc.w    0       sink
        dc.w    0       sblob *
        dc.w    0       spatt *
        dc.w    0       help
        dc.w    336     xsize
        dc.w    224     ysize
        dc.w    infw0-* pinfo *
        dc.w    litm0-* plitem *

        dc.w    app_list0-* pappl *
        dc.w    16384   xsize
        dc.w    12      ysize
        dc.w    0       pinfo *
        dc.w    litm1-* plitem *
        dc.w    0       pappl *
        dc.w    -1

; Sizes
ww0_0   equ     532
ww0_1   equ     148

; Status Areas
wst0    ds.b    71
wst0_e  ds.b    0
        ds.w    0
```

End Of Chapter 31

So, that's the end of the first part of creating a potentially useful utility running under the Pointer Environment. The next article will continue from where we left off and add some code.

Glossary of Abbreviations and Terms Part 3 - H to I

by Dilwyn Jones and Lee Privett

We continue here from where we ended last issue.

Handshaking	When extra control line(s) are used between devices to start, stop and regulate the flow of data.
Hash	The "#" symbol used to indicate a channel number, e.g. in PRINT #0,"Hello"
HD	(i) Hard Disk (ii) High Density, a type of floppy disk or its disk drive
HDD	Hard Disk Drive
HDMI	High-Definition Multimedia Interface, a high quality multimedia connection and interface for audio and video signals using one multicored cable
Heap	Name given to a fairly general storage area used by the operating system or in some cases a user's program running on the computer.
Hexadecimal	Often abbreviated to "hex", this is base 16 arithmetic where each digit is a value from 0 to 15, rather than 0-9 in the decimal system, with the numbers from decimal 10 to decimal 15 expressed as the letters A to F respectively. Because computers work in bits and bytes, this is a convenient numbering system as groups of 4 binary digits can be shown as one hexadecimal digits, and bytes can be expressed as 2 hexadecimal numbers, e.g. decimal 255 is the same as hexadecimal FF. In SBASIC hexadecimal numbers are preceded by a "$" character, such as $FF, while on the PC for example it is common to precede hexadecimal numbers by "0x" e.g. 0xFF
Hermes	Not an abbreviation, this is the name for a replacement for the 8049 second processor in an original QL. It is sold by TF Services, and is designed to improve the handling of the keyboard, serial ports and so on.

High Colour	System used for displaying more than the usual number of QL colours on the screen. Usually used to refer to so-called 16-bit colour, which means that 16 bits of computer memory are used to store the colour value for a single pixel of the display.
High Resolution	When the screen on a QL compatible system is able to display more than the number of pixels possible with a standard QL (more than 512x256). For example, a QL emulator able to show a QL display 800 pixels wide by 600 pixels high might be described as High Resolution.
HOT_REXT	Part of the Pointer Environment (or Extended Environment). This file controls the Hotkeys (see below), and provides a number of new words for the BASIC language, allowing control of hotkeys to start programs, or perform specific actions independent of the program you are using at the time. For example, you can define a hotkey which when pressed would start a copy of Quill whatever you were doing at the time.
HOTKEY	See HOT_REXT above.
HTML	Hyper Text Markup Language (or Hyper Text Meta Language in the USA). A name for a language used to create pages for the World Wide Web.
I/O	Input/Output, or getting information in and out of a computer.
IDE	Intelligent Drive Electronics or Integrated Drive Electronics. A method of connecting drives to computers, where the main interface electronics are part of the drive rather than the computer circuit board. IDE can also stand for Integrated Development Environment, where all programming tools for a task are brought together into one, rather than for example having to load an editor, type in a program, save the program, run a compiler, and link files into one application.
I2C	The bus system used by Minerva Mk 2 from TF Services
INT	Interrupt or Integer. An interrupt is a signal to a microprocessor within a computer that occurs on a regular basis, normally 50 or 60 times a second, or from time to time as required. It means that something is demanding attention and time from the processor, requesting that the processor suspends what it's doing and diverts to whatever device or routine that needs the attention. An integer is a whole number, one that cannot have any decimal places.
Internet	The name given to the global computer network connected by a modem
Interpreter	An interpreter is a computer program that reads the source code of another computer program and executes that program. Because it is interpreted line by line, it is usually a much slower way of running a program than one that has been compiled but is easier for learners because the program can be stopped, modified and rerun without time-consuming compiles each time you make a small change.
IQLR	International QL Report, a QL magazine published by Seacoast Services in the USA and edited for many years by the publisher, Bob Dyl. The magazine eventually ended in 1996, and was superseded by the magazine QL Today.
IQLUG	The original name for Quanta, the QL user group, when it was first set up in 1984. The letters stood for Independent QL Users Group. After a while, they decided it wasn't the easiest of names to pronounce and changed the name of the group to be the same as that of the group's newsletter, QUANTA.
ISA	Industry Standard Architecture, an old style of adapter card used by QXL
ISDN	Integrated Services Digital Network. Basically a posh name for a digital telephone network
ISO	International Standards Organisation

QL Today Volume 17 Subscription/Renewal

This was the last issue of Volume 16 and Volume 17 is coming up soon! The issue 1 of Volume 16 came with a DVD (included in the price) containing all previous issues of QL Today in PDF format - from Volume 1 to Volume 15, English and German (as long as German ones were produced). We are thinking about about another "goodie" next volume, but this also depends on the renewal situation. Saving on sending out reminders gives us extra pages to add to the magazine. So, if you have not renewed yet, please do so.
You can subscribe by using this form (or a copy of it), or subscribe online via www.QLToday.com (the form can be downloaded), by email, letter, fax etc. ... we are flexible!

I hereby subscribe to **QL *Today* for 4 issues of Volume 17. The total price for all four issues** is as follows, including postage and packing (depending on destination)

Destination	price	
☐ **Germany**	**EUR 30.90**	
☐ **Rest of Europe**	**EUR 33.90**	**or £29.90 (UK)**
☐ **Rest of World**	**EUR 38.90**	

☐ Please charge my credit card: ☐ VISA ☐ MasterCard ☐ Diners Club

☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Expires ☐☐ - ☐☐

Card Verification Code: ☐☐☐

☐ Money transfer to one of the following accounts:
- ☐ Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43
- ☐ Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000
- ☐ Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000
- ☐ The Netherlands: Jochen Merz, Gironummer 3258439, Postbank NL Amsterdam
- ☐ and from all other countries in EUR with IBAN and BIC to account
  Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360
- ☐ UK customers can pay £29.90 (price based on exchange rate at print time, valid until August 2012) to
  Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
  or send cheques in £ - no fee for UK sterling cheques (payable to *Jochen Merz* only)!

☐ Payment via Paypal: Log into your paypal account and send the money (in EUR) to **paypal@J-M-S.com**

Name: _____

Street: _____

Town: _____

City: _____

Country: _____

EMail: _____

Date, Signature _____

Please fill in and send to Jochen Merz Software, Kaiser-Wilh-Str. 302, 47169 Duisburg, Germany.
or Fax to +49 203 501517 or scan & Email to SMSQ@J-M-S.com